Guía fácil

Subrutinas útiles en BASIC

Ian R. Sinclair

PARANINFO S.A.

Guía fácil
de

SUBRUTINAS UTILES EN BASIC

IAN R. SINCLAIR

Guía fácil de SUBRUTINAS UTILES EN BASIC

1986

PARANINFO S.A.

MADRID

Traducido por:
RAFAEL CHARFOLE ORENES

© Butterworth & Co. (Publishers) Ltd. England
© de la edición española Editorial Paraninfo, S.A. Madrid (España)
© de la traducción española Editorial Paraninfo, S.A. Madrid (España)

Título original:
Some Useful BASIC Subroutines

Reservados los derechos de edición para todos los países de lengua española. Ninguna parte de esta publicación, incluído el diseño de la cubierta, puede ser reproducida, almacenada o transmitida de ninguna forma, ni por ningún medio, sea éste electrónico, químico, mecánico, electro-óptico, grabación, fotocopia o cualquier otro, sin la previa autorización escrita por parte de la Editorial

IMPRESO EN ESPAÑA
PRINTED IN SPAIN

ISBN: 0-408-01163-7 (edición inglesa)
ISBN: 84-283-1465-9 (edición española)

Depósito Legal: M-7849-1986

PARANINFO S.A. Magallanes, 25 - 28015 Madrid (03304/34/92)

ALCO, artes gráficas. Jaspe, 34 - 28026 Madrid

Indice de materias

	Página
Prefacio	7
1. Presentación visual	13
2. Técnicas de entrada	27
3. Búsqueda y clasificación	43
4. Operaciones con vectores y matrices	61
5. Gráficos y diagramas	73
6. Archivo de datos en casete	87
Apéndice A: Códigos de Control	97
Apéndice B: Subrutinas en código máquina	99

Prefacio

La mayor parte de los programas de ordenadores, a pesar de notables diferencias, usan solamente un número limitado de operaciones con los datos. Algunas operaciones, particularmente en los programas para aplicación científica o de negocios, se usan en todos los programas y muy a menudo, por lo que se preparan como subrutinas.

Si estas rutinas se pueden estandarizar, la tarea de escribir programas se puede reducir frecuentemente a escribir un menú de elecciones, el número de la línea de la subrutina y unas pocas instrucciones más para el usuario del programa. Así, el programador se preocupa solamente de la parte que necesita más habilidad y conocimiento para programar, que es la de desarrollar un plan efectivo y eficiente para el programa y el diagrama de flujo, realizándolo de tal modo que el usuario disfrute ejecutándolo y al mismo tiempo resulte tan libre de errores como sea posible.

El problema de escribir una serie de subrutinas en BASIC es que este lenguaje no está fosilizado sino que al contrario cambia y se desarrolla. Se hace necesario entonces escribir estas subrutinas si ello es posible en un modo simple de Basic, utilizando aquellas instrucciones que se encuentran en los pequeños microordenadores, algunos de los cuales se diseñaron cuando el BASIC era una novedad aún para las máquinas pequeñas.

Pero solamente esto no garantiza que estas subrutinas se ejecutarán sin alteraciones en todos los ordenadores, por lo que el propietario de la máquina deberá hacer cambios cuando sea necesario. Por ejemplo, la instrucción CLS se ha utilizado en las subrutinas como una instrucción para borrar la pantalla y llevar el cursor al origen, pero los usuarios de las máquinas 380 Z y 480 Z tendrán que reemplazarla por la instrucción equivalente PRINT CHR$ (31). Del mismo modo, donde se haya utilizado el punto y coma para continuar imprimiendo en la misma línea (para las

máquinas que usan un tipo de BASIC MICROSOFT) los propietarios del ACORN ATOM deberán omitir el punto y coma, e insertar uno donde no lo haya en las subrutinas, después de una instrucción PRINT. Algunas máquinas no permiten las variables con doble letra como XX$ de modo que en aquellas subrutinas donde aparezcan tales variables se deberán reemplazar por variables de una letra únicamente.

Una dificultad mayor es que algunas funciones no están estandarizadas; el archivo por casete es muy diferente de una máquina a otra, y las operaciones con discos varían tanto, que el archivo por medio de discos no se ha considerado en absoluto en esta obra. Las capacidades gráficas son tan diversas entre las distintas máquinas que solamente se han tratado aquí muy ligeramente, ya que las subrutinas están pensadas para los usuarios que se dedican a cálculos científicos, de negocios o educacionales, más bien que para los que se dedican a juegos.

El método que he usado para resolver estas dificultades ha sido el de escribirlas en BASIC MICROSOFT, pero con un comentario completo sobre cómo opera dicha subrutina, de modo que el usuario pueda utilizar y adaptar la subrutina a su conveniencia. Esto inevitablemente reduce el número de subrutinas que se pueden describir aquí; pero aumenta la utilidad de cada una de ellas.

A causa de la dificultad de reproducir programas, incluso subrutinas cortas con el 100% de precisión por otro método, las muestras de los programas que se dan aquí son partes de programas que ya han funcionado. Esto implica que se han utilizado en alguna máquina en particular (mi propia TRS-80 Modelo 1 Nivel 2). Los comentarios que se incluyen con cada rutina debieran sin embargo permitir al usuario adaptarla a otras máquinas. La mayor parte de las máquinas usan alguna forma del BASIC MICROSOFT de modo que se necesitará muy poca adaptación (o ninguna) en tales casos excepto donde el BASIC sea muy reducido y no tenga las instrucciones del 12K MICROSOFT. Un ejemplo típico es la instrucción PRINT@ y donde esto suceda he ofrecido algunas alternativas. Para mayor claridad cada línea contiene una instrucción solamente, aunque se puede ahorrar mucho tiempo y espacio de memoria si las subrutinas se escriben utilizando líneas con instrucciones múltiples. He usado muy pocas líneas REM porque es más simple comentar las rutinas separadamente, pero las pocas que se usan resultan muy útiles.

El libro se divide en capítulos y cada uno de ellos trata con un juego de tipos de subrutinas; una omisión notable es la de un capítulo que trate con filtros o trampas las secciones de los programas que evitan que el usuario utilice palabras o cantidades que producirían resultados ridículos o paradas por error. La razón para su omisión es que las rutinas de filtro se han

incluído en varias de las rutinas de entrada y no tendría significado separar los tópicos entre sí ya que el filtraje debe hacerse en el momento de introducir los datos.

El modo de dividir el libro en capítulos se refleja en el sistema de numeración de las subrutinas. Cada rutina se identifica por cinco dígitos, el primero (1) es común para todas, el segundo es el número del capítulo, los últimos tres dígitos forman el número de la primera línea de la subrutina; así 13070 es una rutina (la octava) en el capítulo 3. El número de la línea final de una subrutina puede superponerse con el número de la primera línea de la próxima. Ya que me parece improbable que se usen todas las subrutinas juntas, esto no se considera una dificultad.

Aunque la mayor parte de las subrutinas se han hecho para cualquier máquina, se han incluído algunos ejemplos que pretenden resolver problemas específicos para una o dos máquinas. En realidad, un gran número de rutinas específicas de una máquina se refieren al TRS-80 y al Video Genie pero ya que también uso el RML 280Z y el ACORN ATOM no me he olvidado de estas máquinas (1). Pero he tratado de mantener tales rutinas específicas al mínimo.

Todas las subrutinas se llaman con la instrucción GOSUB (número de línea) y se terminan con RETURN. Será necesario para el programa principal que llama a la subrutina traspasarle valores, que son los que tienen que darse a las variables de la subrutina. Cada comentario determina los valores de la variable que se deben pasar de este modo. Se debe ser cuidadoso para evitar problemas entre nombres de variables usados en las subrutinas y los que usan en el programa principal, de aquí la importancia de una cuidadosa planificación del programa. Algunas subrutinas retornarán valores de variables al programa principal.

Es útil recordar que llamar una subrutina significa usar espacio de memoria y tiempo, por lo que es eficiente sólo si se llama varias veces desde distintos lugares del programa principal. En algunos casos se puede preferir construir algunas subrutinas en el programa principal en vez de usarlas como subrutinas propiamente dichas. El uso de GOTO tan despreciado por los académicos es asimismo útil si se desea mantener el texto de la subrutina separado y utilizarlo solamente desde un lugar en el programa; si se hace así se debe reemplazar la sentencia RETURN por otra GOTO que lleve al lugar correcto de retorno.

Siguiendo cada rutina hay una lista de nombres de variables que se dividen en tres grupos. El grupo con la etiqueta PASADAS consiste en

(1) También se incluirán algunos comentarios relativos al IBM PC y sus BASICs de Microsoft. (N. del Ed.)

nombres de variables cuyos valores se pasan a la subrutina. Estos valores se habrán producido o utilizado en la rutina principal y los nombres de las variables de este grupo deben tener valores asignados antes de que se use la subrutina. El grupo con la etiqueta LOCAL consiste en nombres de variables que tienen valores asignados y utilizados dentro de la subrutina y que pueden ignorarse por la rutina principal. El tercer grupo denominado DEVUELTAS consiste de variables cuyos valores son asignados y reasignados dentro de la subrutina y que más tarde se usarán por la rutina principal.

En general, las variables que se pasen a la subrutina pueden también devolverse, pero no es importante volver a nombrarlas.

Los usuarios del microordenador BBC podrán clasificar las variables locales usando la instrucción LOCAL, de modo que los mismos nombres de las variables puedan utilizarse fuera de la subrutina sin que aparezca ningún conflicto por su uso dentro de la subrutina (2).

Cualquier convenio en nombrar las variables no es probablemente útil más que a un pequeño grupo de usuarios, pero en este texto se utiliza el siguiente esquema:

(I) J,K,L,M,Q. Contador de uso general, variables numéricas y de matrices. I se usa para matrices solamente para evitar confusión con 1. Cuando dichos números se ponen en forma de cadena se utiliza el mismo nombre de la variable (ej. K y K$) pero hay que comprobar que el ordenador lo permite.

P,N. Usadas para los índices de elementos de vectores o matrices.

W,X,Y,Z. Variables para elementos de tablas y para contadores.

A,B,C
A$,B$,C$ — Variables numéricas y de tiras de caracteres en la subrutina.

Advertir que las máquinas que permiten el uso de funciones definidas (DEFFN) pueden emplear muchas de estas subrutinas en tal forma, y los usuarios de la máquina BBC tienen también la opción de convertir las subrutinas a procedimientos.

Espero que el libro servirá como una referencia útil para los usuarios del BASIC, especialmente para los nuevos, y proporcionará algunas ideas que cada uno pueda desarrollar cuando las necesite.

Los editores reconocen la ayuda de Clyde Williams y Darryl d'Sousa of the Tandy Corporation en la preparación de este libro.

---

(2) Es decir que las variables locales de una subrutina no tienen nada que ver con las del resto del programa. (N. del Ed.)

1
Presentación visual

Una función común a todos los programas para ordenadores es la representación visual de información. Este primer capítulo se dedica al tema de la presentación gráfica de dicha información en varias formas.

Se podrían escribir muchísimas subrutinas para la representación gráfica de la información, tantas que llenaríamos un libro mucho mayor que éste, por lo que solamente se ha incluído una selección en la que el lector puede hacer numerosas variaciones.

PRINT@

La instrucción PRINT@ que se utiliza en varias versiones de BASIC MICROSOFT tiene la ventaja de permitir que la impresión arranque en cualquier punto de la pantalla, PRINT@ viene seguida por un número que especifica la posición de la pantalla al asignar una numeración consecutiva a lo largo de cada línea.

Como ocurre en el TRS-80 que tiene una línea de 64 caracteres una instrucción PRINT@ que use números de 0 a 63 hará que se imprima en las posiciones a lo largo de la línea superior de la pantalla, de modo que PRINT@ 31, "CENTRO" imprime la C en la mitad de la línea superior. Los caracteres de la siguiente línea se numeran secuencialmente empezando en 64 y terminando en 127 de modo que PRINT@ 95 "CENTRO" colocará la primera letra en el centro de la segunda línea.

El máximo número que puede seguir a PRINT@ en la versión TRS-80 es 1.023 y cuando la instrucción se usa para colocar el texto en la línea inferior, el texto debería continuarse por un punto y coma para evitar el avance de línea.

Notar que la sintaxis requiere un número que siga inmediatamente al símbolo @ con una coma para separar el final de la posición PRINT@ del texto.

Una alternativa elegante es la instrucción TAB (XY) que se utiliza en el ordenador BBC. En este sistema el origen del texto está en la esquina superior izquierda (0,0) X es el número de las posiciones de TAB a través de la pantalla (39 para la anchura de 40 caracteres de MODE 7) e Y es el número de líneas de arriba abajo en la pantalla (0 a 21 en MODE 7).

Distintos modos de operaciones requerirán el uso de números distintos; TAB (x,y) es un método más simple de usar, porque es más fácil localizar una posición en términos de coordenadas XY que por una numeración correlativa dentro de las líneas de la pantalla.

11000 Impresión de un título centrado

El título es asignado a la variable A$ en el programa principal y la variable X debe indicar el máximo número de caracteres por línea para la pantalla del ordenador (un número distintó, normalmente 80 se necesitará para la salida de una impresora).

El número de caracteres por línea es normalmente 40, aunque algunos ordenadores usan 32 y otros 64. La subrutina centrará el título en la parte superior de la pantalla, siempre que la longitud del título no exceda del número de caracteres por línea.

El método utilizado es el clásico de contar el número de caracteres en el título por medio de la instrucción LEN (A$), sustraer este número del total de caracteres permitido por línea y luego dividir la diferencia por 2 para usarlo como el argumento o número de la función TAB para el primer carácter del título. Si hay que centrar un título largo se debería dividir en dos secciones y solicitar dos veces la subrutina. Por ejemplo:

```
100 A$ = "ESTO ES UN INTENTO PARA ESCRIBIR"
110 GOSUB 11000
120 A$ "UN TITULO MUY LARGO"
130 GOSUB 11000
```

Se debe tener en cuenta que CLS significa, borrar la pantalla y llevar el cursor al punto de arranque (rincón superior izquierdo). En algunos orde-

nadores esto se logrará por medio de la instrucción PRINT CHR$(n). El RML-380Z utiliza n=31, el ACORN ATOM n=12, por lo que se deberá consultar la lista de caracteres de control para el ordenador propio si no está incluído en este grupo. El Apéndice A contiene una lista de los caracteres de control.

```
11000 CLS: REM INSTRUCCION PARA BORRAR PANTALLA
11001 PRINT TAB ((X-LEN (A$))/2)A$
11002 RETURN
```

Variables

PASADAS	X número de caracteres por línea
	A$ título
LOCALES	ninguna
DEVUELTAS	ninguna

11010 Título intermitente (versión 1)

Las mismas variables A$ para el título y X para el número de caracteres por línea se pasan a esta rutina. El título se imprime centrado y la GOSUB 11021 en la línea 11013 pide un breve intervalo que decide el ritmo o tasa de intermitencia. El valor 150 en la línea 11021 puede ser alterado para obtener una tasa de intermitencias adecuada, ya que la tasa depende del gusto personal tanto como del ritmo del reloj del ordenador y de la velocidad de ejecución del BASIC. El bucle que empieza en 11014, borra el título, en este caso usando un carácter de control para el retroceso del carro y borrado.

El carácter numérico que se usa CHR$ (8) en la línea 11015 es el utilizado por una gran mayoría de ordenadores pero algunos, principalmente los que no usan códigos ASCII, requerirán un número distinto en la instrucción CHR$. Después de borrar, se incrementa un contador Z, se solicita de nuevo el tiempo de retraso y se repite el proceso hasta que el contador alcanza el límite de 20 en la línea 11019. El número de intermitencias 20 puede naturalmente cambiarse.

```
11010 CLS: REM INSTRUCCION PARA BORRAR LA PANTALLA
11011 Z = 0
11012 PRINT TAB ((X-LEN(A$)/2))A$;
11013 GOSUB 11021
11014 FOR LEN TO LEN(A$)
11015 PRINT CHR$ (8)
11016 NEXT N
11017 Z = Z + 1
11018 GOSUB 11021
11019 IF Z<20 THEN GOTO 11012
11020 RETURN
11021 FOR Q = 1 TO 150
11022 NEXT Q
11023 RETURN
```

Variables

PASADAS	X número de caracteres por línea
	A$ título
LOCALES	N,Q,Z, contadores
DEVUELTAS	ninguna

11020 Título intermitente (versión 2)

Esta es una rutina muy similar a la anterior pero evita el uso del bucle con el carácter de retroceso. La mayor parte de los ordenadores tienen caracteres de control que ejecutarán acciones tales como el borrado hasta el final de la línea y el transporte del cursor al comienzo de la línea. Los caracteres usados aquí son los correspondientes al TRS-80/Video Genie. CHR$ (29) colocará el cursor al comienzo de la línea y CHR$ (30) borrará hasta el final de la línea. El uso de estos caracteres de control ahorra tiempo y esfuerzo de programación. La mayor parte de los ordenadores los poseen de una forma u otra; frecuentemente se accede a ellos a través del teclado usando una tecla CTRL. Nótese que unos pocos ordenadores requerirán una segunda instrucción para llevar el cursor al comienzo de la línea.

```
11020 CLS: REM INSTRUCCION PARA BORRAR LA PANTALLA
11021 Z = 0
11022 PRINT TAB ((X-LEN(A$)/2))A$;
11023 GOSUB 11029: REM RETARDO
11024 PRINT CHR$ (29); CHR$ (30);
11025 GOSUB 11029
11026 Z = Z + 1
11027 IF Z<20 THEN GOTO 11022
11028 RETURN
11029 FOR Q = 1 TO 150
11030 NEXT Q
11031 RETURN
```

Variables

PASADAS	X número de caracteres por línea
	A$ título
LOCALES	Q,Z, contadores
DEVUELTAS	ninguna

11030 Impresión en columnas

El colocar los resultados en columnas puede necesitar una gran cantidad de programación a menos que el ordenador permita el ajuste del tamaño del campo (ACORN ATOM, RML 380Z, BBC Micro computer) de modo que las comas separando las variables tabulen automáticamente en tantas columnas como se necesiten. Esta subrutina dividirá la anchura de la pantalla en un cierto número de columnas usando el número asignado a la variable Y. La variable X igual que antes tendrá asignado el número de caracteres por línea y el vector A$ se usa para los títulos encabezamientos en la parte superior de las columnas, Q es el número de elementos que se imprimirán de este modo y el vector B es un conjunto de los números que se imprimirán en las columnas ya que esta forma de visualización se utiliza generalmente para números.

Es innecesario añadir que el vector de tiras de caracteres puede también imprimirse en columnas.

El programador deberá asegurarse que la longitud de los elementos no exceda la anchura de la columna en este caso.

```
11030 LET Z = INT (X/(Y+1))
11031 FOR N = 1 TO Y
11032 PRINT TAB (N*Z) A$(N);
11033 NEXT N
11034 PRINT
11035 FOR J = 1 TO Q—1 STEP Y
11036 FOR N = 1 TO Y
11037 PRINT TAB (N*Z)B(J+N);
11038 NEXT N
11039 PRINT
11040 NEXT J
11041 RETURN
```

Variables

PASADAS	Q número total de elementos a imprimir
	X número de caracteres por línea
	Y número de columnas
	B() tabla de los números que se van a imprimir
	A$ () tabla de títulos para las columnas
LOCALES	Z,J,N contadores
DEVUELTAS	ninguna

11040 Evitar el desplazamiento

En ocasiones se necesita mantener en la pantalla ciertos datos para referencia mientras que se cambian o elaboran otras informaciones o la pantalla debe mantenerse libre de perturbaciones. En cualquier caso es útil una rutina que evite el desplazamiento. Esta subrutina describe un método basado en el uso de caracteres de control; en el ejemplo se usa el código para mover el cursor hacia la parte superior de la pantalla. Para el TRS 80 este es CHR$ (27), CHR$ (29) es la instrucción que coloca al cursor al comienzo de la línea y CHR$ (30) la que borra hasta el final de la línea. Se deberán reemplazar los números de control por los adecuados cuando se utilicen otras máquinas.

```
11040 PRINT "IMPRIMA ESTO AQUI"
11041 PRINT "Y ESO ALLI"
11042 GOSUB 11046: REM RETRASO
11043 PRINT (CHR$ (27); CHR$ (27), CHR$ (30); CHR$ (29);
11044 "Y LO SIGUIENTE AQUI"
11045 RETURN
11046 FOR Q = 1 TO 500
11047 NEXT Q
11048 RETURN
```

Variables

PASADAS	ninguna
LOCALES	Q contador para el retraso
DEVUELTAS	ninguna

11050 Visualización enmarcada (1)

Un título bien centrado puede parecer todavía mejor si está enmarcado o encerrado en una caja. Esta rutina utiliza un bucle que imprime una línea horizontal en la pantalla encima y debajo del título. Se usan admiraciones en los laterales para los lados del marco. No se ha intentado centrar el título; el método más fácil es definir A$ con las admiraciones y el título, así

A$ = "¡ESTO ES EL TITULO!"

y usar la subrutina 11000 para imprimirlo centrado. Las líneas superior e inferior del rectángulo o caja pueden empezarse con el mismo número en TAB y el tamaño del número del bucle se cambia para hacer el marco tan ancho como se desee.

---

(1) Vale la pena examinar con atención el conjunto de caracteres de la máquina empleada, ya que suelen tener caracteres útiles para dibujar rectángulos.

Así, por ejemplo, en el IBM - PC se tienen los caracteres:

* Angulo superior izquierdo y derecho: 218, 191
* Angulo inferior izquierdo y derecho: 192, 217
* Intersección vertical - horizontal (es decir 4 posiciones de una T): 180, 193, 194, 195
* Una especie de más ampliado: 197
* Línea horizontal continua: 196
* Línea vertical continua: 179

(N. del Ed.)

Recuerde que si existe un carácter gráfico adecuado puede usarse para los lados del rectángulo y un carácter de barra horizontal para las líneas superior e inferior del rectángulo (capítulo V). A la variable X se le asigna el número de caracteres por línea. Haga la línea 11050 del programa de acuerdo con sus requisitos, o reemplácela por una instrucción REM y asigne A$ en otro lugar.

```
11050 Z = LEN (A$)
11051 GOSUB 11056
11052 PRINT
11053 PRINT "!";A$,"!"
11054 GOSUB 11056
11055 RETURN
11056 FOR J = (X-Z)/2 TO (X+Z)/2
111057 PRINT "—";
11058 NEXT J
11059 RETURN
```

Variables

PASADAS	X número de caracteres por línea
	A$ título que hay que imprimir
LOCALES	Z,J contadores
DEVUELTAS	ninguna

11060 Indicación intermitente

En varios puntos en un programa, es útil tener una señal que recuerde que hay que ejecutar una operación determinada, tales como "oprimir cualquier tecla para proseguir" u "oprimir la tecla de puesta en marcha del grabador".

Entonces se necesita una pausa de modo que el operador pueda reaccionar y un asterisco intermitente, por ejemplo, que indique al operador que la máquina espera una entrada. Esto está resuelto en algunas máquinas, pero en aquéllas que no tienen esta posibilidad, esta subrutina proporciona un método de obtener un asterisco intermitente o cualquier otro carácter o incluso el título.

El asterisco se imprime con el punto y coma para obtener la impresión en una línea y la línea 11061 utiliza la función INKEY$ para devolver un valor si se oprime cualquier tecla.

Los ordenadores que no disponen de la función INKEY$ utilizan la GET$. Si no se oprime una tecla se produce una cierta demora o retraso, luego se borra el asterisco utilizando el carácter de espacio de retroceso que es ASCII 8 en la mayor parte de los ordenadores. Otro retraso idéntico es usado entonces antes de que la instrucción GOTO 11060 en la línea 11065 haga que se repita el proceso.

Esta es una subrutina muy útil porque la línea 11060 puede modificarse a PRINT A$ y la línea 11063 cambiada a un bucle.

```
FOR J = TO LEN (A$)
PRINT CHR$ (8)
NEXT J
```

para permitir que los títulos sean intermitentes lo mismo que los asteriscos. La tasa de intermitencia viene controlada por el número en el bucle que determina el retraso en la línea 11066.

Se puede utilizar aquí una variable que permita diferencias en el ritmo de intermitencia, rápido para el asterisco y más lento para el título.

```
11060 PRINT "*";
11061 IF INKEY$< > "" THEN RETURN
11062 GOSUB 11066: REM RETARDO
11063 PRINT CHR$ (8);
11064 GOSUB 11066
11065 GOTO 11060
11066 FOR Q I = 1 TO 100
11067 NEXT Q
11068 RETURN
```

Variables

PASADAS	ninguna
LOCALES	Q contador
DEVUELTAS	ninguna

11070 Una pantalla a la vez...

Algunos ordenadores permiten formar páginas en la pantalla lo que significa que cuando una colección de datos se imprime, la pantalla se llenará, pero la impresión se detendrá hasta que se oprima cualquier tecla. Cuando esta facilidad no existe, solamente una rutina con código de máquina puede utilizarse para formar "páginas", pero la impresión de los datos de un programa puede fácilmente hacerse en "páginas" utilizando esta subrutina.

La variable J se usa para el número de líneas por "página", la subrutina supone que se van a imprimir un número de frases igual a Z, almacenadas en el vector A$; recordar el uso de la subrutina anterior para hacer intermitente un asterisco al final de cada cuadro.

```
11070 FOR N = 1 TO Z
11071 PRINT A$ (N)
11072 IF INT (N/J-1)=N/(J-1) THEN GOSUB 11060
11073 NEXT N
11074 RETURN
```

Variables

PASADAS	J líneas por página
	Z número máximo de líneas a escribir
	A$ () Vector de datos
LOCALES	N contador
DEVUELTAS	ninguno

11080 Contaje de datos introducidos

Uso varios programas que leen los datos de los archivos de las casetes. Esta subrutina cuenta cada entrada de la casete de modo que se puede ver mirando la pantalla, hasta donde se ha llegado en ese momento. Disminuye por tanto la ignorancia de la situación en lo que a archivos de casete se

refiere, y es particularmente útil para máquinas tales como la TRS80 y Video Genie que no dan indicación, salvo la presencia de un asterisco, de que se están leyendo los archivos.

```
11080 CLS
11081 FORN N = 1 TO X
11082 GOSUB 11086: REM CARGA DE LA CASETE
11083 PRINT; "";
11084 NEXT N
11085 RETURN
```

Variables

PASADAS	X número de ficheros a carga
LOCALES	N contador
DEVUELTAS	ninguno

11090 Clasificación de cuentas

Esta es una técnica que no es fácilmente transferida del TRS 80 a otros ordenadores y se ha dejado en su forma original comprimida, en vez de volver a escribirla en líneas con una instrucción en cada una.

Las líneas 10 a 30 indican el tipo de datos que se necesitan para preparar la subrutina.

El principio en que se basa es que los programas de cuentas usan dos columnas, una para las entradas y otra para las salidas, pero no existen necesariamente datos para cada uno de las partidas de entrada y salida.

Esta subrutina consigue disponer ambas columnas nítidamente cuando no se necesitan todos los elementos.

Los espacios no se imprimen. La técnica se basa en el uso de la instrucción PRINT@ que no existe en todos los ordenadores —incluso algunas versiones de RML-380Z BASIC omiten esta instrucción—.

La nitidez que produce la subrutina y la economía de espacio de la pantalla compensa el tiempo que gaste en escribirla.

Obsérvese el uso del avance de renglón en 11093 y la superflua instrucción PRINT en 11094 que permite espaciar antes de subrayar.

```
5      CLEAR 500
10     N3=5;A1(1)=23:A1(3)=24:A$(1)="VENTAS"
       A$(3) "OTROS": A2(2) = 45
       A2 (4) = 33: B$(2) = "LICOR":B$(4) = "CAJA MENOR"
20     CLS: GOSUB 11090
30     PRINT STRING$ (64, "—"): END
11090  PRINT TAB (10) "INGRESOS"; TAB (35)
       "SALIDAS": PRINT
       GOSUB 11091: RETURN
11091  K1=0: J=128: FOR N=1 TO b N3:IF A1(N) <>0 THEN
       PRINT@
       J,A$(N);TAB(17)A1(N);:J=J+64:K1=K1+1:NEXT N:
       ELSE NEXT N
11092  K2=0:J=160: FOR N=1 TO b N3: IF A2(N) < >0
       THEN PRINT@ J,B$ (N);
       TAB (50) A2(N); : J=J+64:K2=K2+1:NEXT N : ELSE
       NEXT N
11093  IF K1 > K2 THEN J=64*(K1+1) ELSE J=64*(K2+1)
11094  PRINT@ J, "": RETURN
```

Variables

PASADAS	N 3 número máximo de elementos en cada columna
	A1() vector de números para la columna 1
	A2() vector de números para la columna 2
	A$() vector de nombres columna 1
	B$() vector de nombres columna 2
LOCALES	J,K,N contadores
DEVUELTAS	ninguna

11100 Título desplazable (versión 1)

Un título que se desplaza consigue frecuentemente más atención que uno estático, y la subrutina para lograr esto es bastante simple. Como de cos-

tumbre se asigna a A$ el título y a X el número máximo de caracteres por línea. El resultado de la rutina es mover el título hacia dentro desde la parte derecha de la pantalla con una tasa controlada por el retraso especificado en la línea 11104. La rutina del retraso no está especificada aquí.

Observar el uso de la instrucción PRINT@ que utiliza 512 como el número que representa la posición final en una línea. Estos números se deberán cambiar para adecuarlos a la configuración del ordenador que se utilice.

```
11100 Z=LEN (A$)
11101 Y=1
11102 FOR N=1 TO X
11103 PRINT@ (512-N),MID$ (A$,1,Y); ""
11104 GOSUB*: REM RETARDO
11105 IF Y<Z THEN GOTO 11108
11106 NEXT N
11107 RETURN
11108 Y=Y+1
11109 GOTO 11106
```

Variables

PASADAS	X número máximo de caracteres por línea
	A$ título
LOCALES	Z,Y,N contadores
DEVUELTAS	ninguna

11110 Título desplazable (versión 2)

En los ordenadores en los que su BASIC no tiene la instrucción PRINT@, esta versión del título movible usa únicamente la instrucción TAB que es aceptable pero menos satisfactoria. Advertir que 11113 utiliza el número 62, esto es X-2, donde X es el número de caracteres por línea y si el ordenador lo permite TAB (X-2-N) puede usarse en esta línea con un valor para X alimentado a la subrutina.

La línea 11110 puede modificarse de acuerdo con sus requirimientos.

```
11110 Z=LEN (A$)
11111 Y=1
11112 FOR N=1 TO X
11113 PRINT TAB (62-N)MID$ (A$,1,Y); ""
11114 GOSUB*: REM RETARDO
11115 IF Y<Z THEN GOTO 11118
11116 PRINT CHR$ (27);:NEXT N
11117 RETURN
11118 Y=Y+1
11119 GOTO 11116
```

Variables

PASADAS	X caracteres por línea
	A$ título
LOCALES	Z,Y,N contadores
DEVUELTAS	ninguna

2

Técnicas de entrada

Si la mitad de nuestro tiempo usando ordenadores se gasta o consume viendo las pantallas, la mayor parte de la otra mitad se debe emplear usando el teclado para la entrada de información.

De nuevo, unas cuantas subrutinas pueden facilitar el trabajo, y en este capítulo se describen un número de métodos de entrada, que van desde los muy simples a los relativamente complicados.

La instrucción INPUT de la que disponen la mayoría de los ordenadores detiene el programa hasta que se hayan oprimido las teclas apropiadas seguido por la instrucción RETURN (o su equivalente) y una interrogación aparece en la pantalla hasta que esto se haya realizado. La mayoría de los ordenadores permitirán una entrada con cualquier tecla cuando se usa la instrucción INPUT X$, pero aceptarán sólo números si se usa INPUT X. Además, la mayor parte de los ordenadores siguen la entrada de datos con un desplazamiento de línea y retorno de carro.

INKEY$

Esta es una instrucción utilizada en varias versiones del BASIC MICROSOFT y otros para solicitar una entrada de una línea sin necesidad de usar la tecla de RETURN. La sintaxis varía algo, pero el método normal es el de asignar una variable al resultado de la operación como en:

```
1000 A$ = INKEY$
```

Tal como está escrito se asignaría un valor solamente si se oprimiera una tecla en el momento de ejecución de la línea, de modo que INKEY$ se escribe generalmente en un bucle tal como

```
1000 A$ = INKEY$: IF A$ "" THEN 1000
```

que mantiene la línea realizando un bucle sobre si misma hasta que se detecta que una tecla ha sido oprimida.

El micro BBC usa la instrucción de un modo distinto con anillo o bucle temporizado incorporado de modo que:

```
1000 A$ = INKEY$ (100)
```

recirculará durante un segundo y luego continuará, tanto si se oprime una tecla o no. Oprimiendo la tecla dentro del tiempo del bucle transfiere el carácter a A$, pero si se agota el tiempo sin oprimir ninguna tecla, se le asigna a A$ el valor "—1". El número entre paréntesis en la línea es el tiempo de retraso en centisegundos. Una instrucción INKEY en la máquina BBC, transfiere el carácter en el código ASCII de la tecla que se oprimió, si no "—1".

La alternativa al uso convencional de INKEY$ es GET$. Esta se usa en varias versiones de MICROSOFT y por la máquina BBC, como el método principal de obtener un carácter del teclado con una sintaxis como:

```
1000 A$ = GET$
```

o

```
1000 IF GET$ = "A" THEN...
```

El microordenador BBC también permite el uso de GET, transfiriendo el código ASCII de la tecla oprimida. Otros ordenadores requieren que GET$ se use en un bucle como con INKEY$ y otros (como el BBC) tienen un bucle incorporado de modo que con GET$ habrá un retraso indefinido hasta que se oprima una tecla.

12000 ¿Cuántos?

Cuando se necesita introducir un cierto número de elementos o items, su número puede tener que especificarse si un bucle del tipo FOR ... NEXT se usa para leer los elementos e introducirlos. Esta subrutina proporciona un método de introducción de dicho número de elementos. La interrogación impresa por el uso de INPUT lo hará en la misma línea que la pregunta. Es más claro que el método usual de dos líneas. Con el ATOM deberán omitir el punto y coma.

```
12000 PRINT "CUANTOS ELEMENTOS";
12001 INPUT X
12002 RETURN: REM AL PROGRAMA BASIC
```

Variables

PASADAS	ninguna
LOCALES	ninguna
DEVUELTAS	X número de items

12010 Entrada en la misma línea

El uso tradicional de INPUT es el de tener una entrada por línea a causa del avance de línea y del retorno del carro que sigue a INPUT debido al uso de RETURN. Esta subrutina permite que dos instrucciones separadas de INPUT estén en la misma línea de la pantalla usando el código de órdenes para subir el cursor que es 27 para el TRS 80 y 145 para PET. Esta orden tiene que usarse con TAB para evitar que se borre la entrada anterior de modo que puede ser necesario alterar el número en TAB en 12012 si la primera entrada es una cadena larga en vez de un número.

```
12010 PRINT "TECLEE UN NUMERO, POR FAVOR";
12011 INPUT X
12012 PRINT TAB (20) CHR$ (27);: REM SUBIR UNA LINEA
12013 PRINT "NOMBRE, POR FAVOR";
12014 INPUT X$
12015 RETURN
```

Variables

PASADAS	ninguna
LOCALES	ninguna
DEVUELTAS	X número
	X$ nombre

12020 Entrada tabulada-versión simple

Una entrada tabulada con cada item o elemento introducido debajo de la cabecera apropiada es mucho más clara que el INPUT convencional con un item en cada línea.

Esta subrutina descrita con un trozo de un programa típico que la llama en las líneas 10-40, permitirá la colocación de un cierto número de elementos debajo del encabezamiento apropiado. A la variable X se le asigna el máximo número de caracteres por línea, el ejemplo utiliza 64 que es el de las TRS-80/Video Genie. Cuando se ejecuta cada entrada, una variable N se encarga de tabular en la posición correcta a la línea 12032 y el carácter para desplazar hacia arriba el cursor (27 en este ejemplo) devuelve la siguiente entrada a la misma línea.

```
10      W = 64:PRINT "ELEMENTOS A LISTAR"
20      GOSUB 12020
30      PRINT "LOS VALORES SON" X$, " ", Y1; " "; Y2
40      END
12020   PRINT "NUMERO"; TAB(X/4) "ELEMENTO"; TAB
        (X/2) "COSTE"; TAB (3*X/4) "PRECIO"
12021   INPUT Y
12022   N = 1
12023   GOSUB 12032:REM ESPACIO LATERALMENTE Y HACIA
        ARRIBA
12024   INPUT A$
12025   N = 2
12026   GOSUB 12032
12027   INPUT Y1
12028   N = 3
12029   GOSUB 12032
12030   INPUT Y2
12031   RETURN
12032   PRINT TAB (N* X/4-2)CHR$(27);
12033   RETURN
```

Variables

PASADAS	X caracteres por línea
LOCALES	N contador
DEVUELTAS	Y número
	A$ nombre de los elementos
	Y1 Coste
	Y2 precio

12030 Entrada Tabulada-versión con bucle

La subrutina de entrada tabulada puede acortarse usando un bucle como se indica en esta versión, y este método es preferible si los intervalos de tabulación son regulares. Para una tabulación irregular se puede usar la rutina anterior pasando N directamente como el número de tabulaciones.

```
12030 PRINT "NUMERO"; TAB (X/4) "ELEMENTO"; TAB
      (X/2) "COSTE"; TAB (3*X/4) "PRECIO"
12031 FOR N = 1TO4
12032 INPUT A$(N)
12033 GOSUB 12036
12034 NEXT N
12035 RETURN
12036 PRINT TAB (N*X/4-2)CHR$ (27);
12037 RETURN
```

Variables

PASADAS	X caracteres por línea
LOCALES	N contador
DEVUELTAS	A$ () vector de entradas

12040 Comprobación de números

Muchos programas requieren que se compruebe el tamaño de un número, por ejemplo la respuesta a una elección de menú. Esta subrutina es un ejemplo simple de tal validación, llamada también menos elegantemente una "trampa". La entrada es una variable de caracteres para así evitar una parada en el caso de que se oprima por error una tecla que no sea numérica. El valor de la variable se extrae para obtener el número si se oprimió una tecla numérica y la línea 12043 entonces comprueba el tamaño del número —ponga aquí sus propios valores—. Advertir que una entrada de 0 no se acepta porque una tecla no numérica puede dar 0 como el valor de la serie. La línea 12044 es importante; el usuario debe ser informado de por qué no se acepta una entrada.

```
12040 PRINT "TECLEE UN NUMERO POR FAVOR";
12041 INPUT X$
12042 X = VAL (X$)
12043 IF X>0 AND X<10 THEN RETURN
12044 PRINT CHR$ (27); TAB (5) "ERROR... EL NUMERO DEBE SER
      DE 1 A 9"
12045 GOTO 12040
```

Variables

PASADAS	ninguna
LOCALES	X número que se comprueba
DEVUELTAS	X$ número de entrada en forma de tira

12050 Oprima cualquier tecla ... (1)

Esta rutina usa INKEY$ en una forma que evita problemas de obtener información inservible, ya que la variable K$ no es asignada hasta que se ha oprimido una tecla. Algunos ordenadores, entre ellos el TRS-80/Video Genie no aceptarán este uso de INKEY$.

```
12050 IF INKEY$ = ""THEN 12050
12051 K$ = INKEY$
12052 RETURN
12053 REM EVITA PROBLEMAS DE LECTURA DE DATOS
      ERRONEOS PERO NO TODOS LOS ORDENADORES
      ASIGNARAN UN VALOR A K$
```

Variables

PASADAS	ninguna
LOCALES	ninguna
DEVUELTAS	K$ valor del carácter de la tecla oprimida

12060 Oprima cualquier tecla ... (2)

Esta es la versión de la subrutina anterior para los micros TRS-80/Video Genie. Tiene la desventaja de que se acumula información inservible, pero esto puede evitarse usando la instrucción

ZZ = FRE (A$)

dentro del bucle. La instrucción FRE (A$) automáticamente provoca una acción de borrado de información inútil, evitando así su acumulación. A$ es una variable ficticia, no necesita que se le asignen valores y la variable ZZ será el número de bytes de memoria libres que no se necesitan.

```
12060 K$ = INKEY$
12061 IF K$ = "" THEN 12060
12062 RETURN
12063 REM ESTA VERSION SE PRECISA EN UN TRS-80 SI SE
      TIENE QUE DEVOLVER UN VALOR DE K$
```

Variables

PASADAS	ninguna
LOCALES	ninguna
DEVUELTAS	K$ valor del carácter de la tecla oprimida

12070 El interludio SI ... NO

El uso continuo de la tecla RETURN necesario cuando se usa INPUT puede ser molesto, aunque puede dar tiempo para pensar mejor lo que se hace.

Esta subrutina permite que se impriman las palabras YES y NO sin el uso de RETURN y pasará un código numérico, 2 para NO y 1 para YES, de vuelta a la rutina a la variable Z, según qué palabra se haya tecleado. Una rutina con trampa (filtro) comprueba o evita errores de impresión en la línea 12080. Las líneas 12071-12072 se pueden reemplazar por una llamada a la rutina con asterisco intermitente si se quiere y el mensaje ERROR puede también hacerse intermitente. Esta rutina es útil porque puede adaptarse a

muchas respuestas distintas (HELP por ejemplo, en el caso de que se necesiten intrucciones) y no es necesario esperar a que se imprima toda la palabra.

```
12070 A$ = ""
12071 K$ = INKEY$
12072 IF K$ = "" THEN 12071
12073 PRINT K$;
12074 A$ = A$ + K$
12075 X = LEN (A$)
12076 IF X<2 THEN 12071
12077 IF X = 2 AND A$ = "NO" THEN 12082
12078 IF X =  2 AND A$ = "SI" THEN 12084
12079 IF X =  2 THEN 12071
12080 PRINT "ERROR: CONTESTE SI O NO"
12081 GOTO 12070
12082 Z = 2
12083 GOTO 12085
12084 Z = 1
12085 PRINT
12086 RETURN
```

Variables

PASADAS	ninguna
LOCALES	X contador de la longitud de la palabra
	K$ carácter de la tecla
	A$ palabra obtenida de K$
DEVUELTAS	Z código numérico (NO = 2, SI = 1

12080 Trampa (filtro) para limitar la entrada (1)

Cuando hay que limitar la longitud de una tira de caracteres en la entrada a las máquinas, en las que la longitud máxima de una tira de caracteres en una matriz o vector está limitada, por ejemplo, puede ser útil una subrutina de este tipo.

En el ejemplo se comprueba que la longitud de la tira de caracteres esté comprendida entre 5 y 10 caracteres en la línea 12082, y se imprimirá un

mensaje de error para las tiras de caracteres fuera de este margen. En muchos casos sólo se necesita rechazar las tiras de caracteres excesivamente largas.

```
10      A$ = "POR FAVOR USE DE 5 A 10 LETRAS"
20      PRINT "TECLEE EL NOMBRE"; A$
30      GOSUB 12080
40      PRINT B$
50      END
12080   INPUT B$
12081   X = LEN (B$)
12082   IF X< = 10 AND X => 5 THEN RETURN
12083   IF X = > 10 THEN 12086
12084   PRINT "NOMBRE DEMASIADO CORTO-"; A$
12085   GOTO 12080
12086   "NOMBRE DEMASIADO LARGO-"; A$
12087   GOTO 12080
```

Variables

PASADAS	A$ mensaje de ayuda
LOCALES	X longitud
DEVUELTAS	B$ nombre aceptado

12090 Trampa para limitar la entrada (2)

Si no se necesitan trampas para una determinada longitud, esta modificación de la rutina anterior permite que el número que indica la longitud se transfiera a la subrutina cada vez que se solicita bajo la forma de dos variables N1 y N2.

```
12090 INPUT B$
12091 X = LEN (B$)
12092 IF X< = N1 AND X => N2 THEN RETURN
12093 IF X => N1 THEN 12096
12094 PRINT "NOMBRE DEMASIADO CORTO-"; A$
12095 GOTO 12090
12096 PRINT "NOMBRE DEMASIADO LARGO-"; A$
12097 GOTO 12090
```

Variables

PASADAS	N1, N2, límites de la longitud.
	A$ Mensaje de ayuda
LOCALES	X longitud de la cadena
DEVUELTAS	B$ salida

12100 Comprobación de números

Cuando la entrada de un cero, un número negativo o un número excesivamente grande pudiera causar problemas (tal como el error de dividir por cero) una subrutina con trampa de esta clase es útil —se advierte que es válida solamente para números—, si se oprime una tecla de letras, el sistema operativo de la máquina rechazará la selección y la devolverá al estado de READY (PREPARADA). Si se quiere evitar esto sobre todo cuando los que usan la máquina no están bien entrenados, entonces se debería usar una tira de entrada seguida por la instrucción VAL (tira) tal como se describe en 12040 (Comprobación de Números).

```
12100 INPUT X
12101 IF X< = 0 THEN 12104
12102 IF X> 9 = THEN 12106
12103 RETURN
12104 PRINT "NO SE PERMITE 0 O NUMEROS NEGATIVOS
12105 GOTO 12100
12106 PRINT "NUMERO DEBE SER MENOR QUE 10"
12107 GOTO 12100
```

Variables

PASADAS	ninguna
LOCALES	ninguna
DEVUELTAS	X sólo números

12110 Bloque de entradas y comprobación visual

No es siempre posible decidir lo que es inaceptable como entrada, de modo que una manera útil de hacerlo es ofrecer al operador a intervalos una lista de entradas para que las compruebe.

Esta subrutina utiliza un bucle en la línea 12110 que acepta entradas en la forma de un vector de caracteres.

El número total de elementos se debe transferir a la subrutina como un valor para la variable Z. La línea 12113 interrumpe el procedimiento de entrada después de cada diez, de modo que se pueda hacer la comprobación de validez.

En la línea 12117 se borra la pantalla (habrá que usar el código individual para el borrado de la pantalla) y entonces se imprime un bloque de diez entradas acompañado de un número de referencia para cada elemento.

La rutina que empieza en 12113 da entonces entonces al operador la oportunidad de cambiar las entradas tantas veces como sea necesario para eliminar errores. Todo el bloque de entradas aparece en pantalla después de cada corrección, y la liberación se hace imprimiendo el 0 en lugar de un número.

```
12110 FOR N = 1 TO Z
12111 PRINT "ITEM NO." N;
12112 INPUT A$ (N)
12113 IF INT (N/10) = N/10 THEN GOSUB 12117
12114 NEXT N
12115 GOSUB 12117
12116 RETURN
12117 CLS
12118 PRINT "ESTE BLOQUE DE ENTRADAS ES-"
12119 FOR J = (N-9) TO N
12120 PRINT TAB (5)J;CHR$(8)".";A$(J)
12121 NEXT J
12122 PRINT
12123 PRINT "TECLEE: NUMERO A MODIFICAR O 0 PARA SALIR;
12124 INPUT Y
12125 IF Y = 0 THEN 12129
12126 PRINT "ITEM"; Y; "DEBE SER";
12127 INPUT A$ (Y)
12128 GOTO 12117
12129 CLS
12130 RETURN
```

Variables

PASADAS	Z número de elementos
LOCALES	N, J, contadores
	Y indicador SI/NO
DEVUELTAS	A$ vector de cadenas de entrada

12130 Rellenar hasta obtener un tamaño previsto, añadiendo datos por el lado izquierdo

Para muchas operaciones, especialmente cuando se usan rutinas que cortan las tiras, todo resulta más simple si cada tira es de la misma longitud. Esta rutina cuenta el número de caracteres de cada tira entrante y si es menor que la longitud considerada la rellenará por el lado izquierdo hasta obtener un total de Y caracteres usando el carácter blanco de ASCII 128.

La variable Y se debe transferir desde el programa principal, o insertada como un número en la subrutina.

Esta subrutina debería estar precedida por una rutina de comprobación de la longitud de la cadena tal como la 12090.

Algunos ordenadores no usan el código estándar ASCII (PET, SHARP MZ 80 y el AZ 81) de modo que no se puede usar el número 128. El carácter blanco que se usa debería ser uno que no esté en el teclado, o sea, no es el espacio en blanco que se produce por la barra de espaciamiento (32 en el código ASCII).

```
12130 INPUT A$
12131 X = LEN (A$)
12132 FOR N = 1 TO (Y-X)
12133 A$ = CHR$ (128) + A$
12134 NEXT N
12135 RETURN
```

Variables

PASADAS	Y máximo tamaño del relleno
	A$ tira que se rellena
LOCALES	X longitud de la tira
DEVUELTAS	A$ salida rellenada

12140 Rellenar hasta obtener un tamaño previsto por el lado derecho y cortar

Esta es una extensión de la rutina anterior que acumula o empaqueta una tira a una longitud determinada por la variable Y utilizando espacios en blancos por el lado derecho. Luego se reduce el tamaño de la tira acumulada

en la línea 12145, en este caso a seis caracteres, pero obviamente esto puede cambiarse. No existe aquí una subrutina para limitar la longitud de la tira de entrada (A$) porque todas las tiras terminarán con el mismo tamaño.

La mayor parte de las máquinas trabajarán bien con la línea 12145 que extrae los primeros seis caracteres de A$ pero el ACORN ATOM y el ZX 81 necesitarán un juego diferente de instrucciones (pág. 42).

```
12140 INPUT A$
12141 X = LEN (A$)
12142 FOR N = 1 TO (Y-X)
12143 A$ = A$ + CHR$ (128)
12144 NEXT N
12145 A$ = LEFT$ (A$,6)
12146 RETURN
```

Variables

PASADAS	Y máximo tamaño del relleno
	A$ tira a rellenar
LOCALES	X longitud de la tira
	N contador
DEVUELTAS	A$ entrada cortada y rellenada a 6 caracteres

12150 Entrada múltiple con comas

La mayor parte de los ordenadores permiten la entrada de más de un valor variable en cada instrucción usando una coma como separador. La coma no se almacena sin embargo y si se insertan comas entre tiras en la memoria, algunas operaciones, principalmente el archivo en casetes, no se pueden hacer. Esta rutina consiste en dos partes. Las líneas entre 12150 y 12152 concatenarán tiras que han entrado separadas usando el espacio en blanco ASCII 128 como un separador entre tiras. La longitud de cada tira tendrá que comprobarse por una subrutina separada (no se da aquí) para asegurarse que la tira concatenada o unida en la línea 1215 no es demasiado grande para que la maneje el ordenador. La mayor parte de las máquinas manejan tiras de hasta 248 caracteres y otras hasta 255.

La segunda parte de la subrutina trata de la separación y de la inserción de la coma. La cadena concatenada se separa examinando cada carácter a la vez e imprimiéndolo si no es el ASCII 128.

Cuando aparece un código 128 la rutina 12159 imprime una coma en su lugar de modo que el aspecto final de la cadena es el mismo que tenía en la entrada con las comas. Se puede hacer una pequeña modificación en la rutina para recobrar la variable A$ y las variables numéricas B y C de la tira concatenada.

```
12150 INPUT A$, B, C
12151 C$ = A$+CHR$ (128)+STR$(B)+CHR$(128)+STR$(C)
12152 RETURN
12153 FOR N = 1 TO LEN (C$)
12154 D$ = MID$ (C$,N,1)
12155 IF D$ = CHR$ (128) THEN 12159
12156 PRINT D$;
12157 NEXT N
12158 RETURN
12159 D$ = ","
12160 GOTO 12156
```

Variables

Primera parte	LOCALES	ninguna
	DEVUELTAS	C$ concatenada con espacios en blanco
	PASADAS	C$ como arriba
Segunda parte	LOCALES	D$ caracteres separados de la tira
	DEVUELTAS	ninguna

12160 Agrupamiento de tiras

Las tiras que forman un vector se pueden procesar más eficientemente si se pueden agrupar en tiras largas de la máxima longitud permitida. Esto es especialmente cierto en los sistemas con archivo de casetes del TRS 80/Video Genie, porque un archivo de datos de casete viene siempre precedido por 256 bytes de ceros independientemente de la longitud de los datos del archivo, con lo que se hace muy tediosa la grabación de un vector de tiras.

Incluso para máquinas con sistemas de archivo de datos más eficiente la rutina de acumulación puede ser útil pues una única tira larga se procesa mejor que un vector.

La rutina permite que cuatro juegos de cadenas agrupadas formen el nuevo vector A$. En la línea 12164 se introducen los elementos de la cadena, y en la línea 12167 se acumulan usando ASCII 128 como un separador. A medida que cada tira se llena, se crea una nueva y agrupada A$, hasta que no queda más espacio disponible. Esta condición está indicada en la línea 12173.

La variable cuyo valor tiene que pasarse a la subrutina es K, la máxima longitud de tira permitida. El valor elegido para K debería ser menor que el máximo absoluto de 255, porque algunas rutinas de ordenador no manejan tiras de longitud máxima. Un ejemplo es el archivo de casetes del TRS 80 que acepta 254 caracteres para grabar pero solamente 248 para lectura.

```
12160 FOR N = TO 4: REM SE EMPLEAN 4 GRUPOS
12161 A$ (N) = ""
12162 NEXT N
12163 J = 1
12164 FOR N = 1 TO 100
12165 PRINT "ELEMENTO"; N;
12166 INPUT B$ (N): REM DEBE DIMENSIONARSE
12167 A$ (J) = B$ (N) + CHR$(128)
12168 IF LEN (A$(J)) = K THEN 12171: REM K ES LA LONGITUD
      MAXIMA DE LA TIRA EMPAQUETADA
12169 NEXT N
12170 RETURN
12171 J = J+1
12172 IF J < = 4 THEN 12169
12173 PRINT "AGOTADO EL ESPACIO. DETENGASE"
12174 IF INKEY$ = "" THEN 12174
12175 GOTO 12170
```

Variables

PASADAS	K máxima longitud de tira permitida
LOCALES	J,N, contadores
	B$ tira de entrada
DEVUELTAS	A$ tiras concatenadas

12170 Desagrupamiento de tiras

Esta subrutina invierte la acción de la anterior convirtiendo las tiras agrupadas en la distribución original de tiras. La línea 12170 utiliza un bucle FOR ... NEXT con 100 elementos; en la práctica este valor sería transferido desde el programa principal o leído desde la cinta como un elemento que precede a las tiras.

El bucle en 12174 también supone un número pasado a la subrutina, en este caso 4 (el número total de cadenas acumuladas) pero este número también sería normalmente pasado desde la rutina principal o desde una lectura de la cinta. La subrutina opera cambiando los caracteres uno cada vez en una distribución de tiras, empezando con un nuevo número N como subíndice cada vez que aparece el carácter ASCII 128. El carácter variable del contador K se utiliza para asegurarse que no se leerán caracteres más allá del final de una tira haciendo que se reporte un error y un retorno a READY.

```
12170 FOR N = 1 TO 100
12171 B$ (N) = " "
12172 NEXT N
12173 N = 1
12174 FOR J = 1 TO 4
12175 K = 1
12176 C$ = MID$ (A$(J),P,1)
12177 IF C$ = CHR$ (128) THEN 12184
12178 B$ (N) = B$ (N)+C$
12179 K = K + 1
12180 IF K => LEN ((A$(J) THEN 12182
12181 GOTO 12176
12182 NEXT J
12183 RETURN
12184 N = N+1
12185 GOTO 12179
```

Variables

PASADAS	A$ tira concatenada
LOCALES	N,J,K, contadores
DEVUELTAS	B$ () vector de tiras

3

Búsqueda y clasificación

Búsqueda y clasificación, además de ser el título de uno de los más fascinantes libros sobre teoría de computación, es un título que trata con algunas de las subrutinas, más interesantes, más frustradoras y más útiles.

Revisando programas que he escrito, hay muy pocos que no tengan algunas de las rutinas que se describen en este capítulo para distintos propósitos.

Todas las operaciones en este capítulo suponen que existe un vector o matriz y que el nombre de su variable se transfiere de la rutina principal junto con el número de elementos en él. Algunas operaciones se suponen que son realizadas en tiras.

La adaptación a vectores de números es simple, pero por muchas razones resulta a menudo mejor usar vectores de cadenas no siendo la menor la facilidad con la que pueden rellenarse hasta un tamaño estándar, acumulados y recortados utilizando instrucciones como LEFT$ MID$ y RIGHT$. Los valores numéricos se pueden convertir en tiras usando la función STR$ que existe en casi todos los ordenadores y extraídos de las tiras usando VAL (nombre de la tira), de modo que no hay ninguna desventaja en utilizar vectores de tiras exclusivamente.

El micro BBC tiene una nueva instrucción muy útil, EVAL, que evaluará el contenido de una tira como si consistiera en números e instrucciones.

Esto hace posible las operaciones aritméticas con tiras sin tener que usar una serie de operaciones VAL en diferentes ocasiones.

Esta es una de las muchas adiciones a BASIC, que hacen que esta máquina sea tan buena, probablemente la contribución más significativa en lo que se refiere a microordenadores.

Seccionamiento de tiras

Las instrucciones de seccionamiento de tiras de dos ordenadores difieren considerablemente del método MICROSOFT de modo que una guía de la traducción de estas instrucciones es útil.

ZX80, ZX81, ZX SPECTRUM

La instrucción de seccionamiento de tiras usa la palabra TO y es de la forma "tira (comienzo TO final)". Esta forma reemplaza elegantemente las instrucciones LEFT$, MID$ y RIGHT$ usadas en MICROSOFT con un formato más simple.

En la tabla 3.1 se dan ejemplos de equivalencias para hacer las conversiones fáciles.

Tabla 3.1. **Ordenes para seccionamiento de tiras. Diferencias entre Microsoft y Sinclair**

Microsoft	Ejemplo	Sinclair	Ejemplo
LEFT$ (tira, número)	B$=LEFT$ (A$,2)	tira (TO fin)	LET B$=A$ (TO2)
RIGHT$ (tira, número)	B$=RIGHT$(A$,3)	tira (arranque TO)	LET B$=A$(3TO)
MID$ (tira, arranque, fin)	B$=MID$(A$,2,6)	tira (arranque TO fin)	LET B$=A$(2TO6)

ACORN ATOM

Las instrucciones de seccionamiento de tiras del ACORN ATOM no son fáciles de entender hasta que no se da uno cuenta que dependen del método o manera que el ACORN ATOM usa para las tiras. Una tira en ATOM BASIC se escribe en la forma $A en vez de en la más conocida A$; utilizando este formato, A es un número que es también el número de la dirección del primer carácter de la tira en la memoria, y la dirección en la memoria de otros caracteres en la tira pueden obtenerse añadiendo el número del lugar a A. Este esquema también se utiliza con el más corriente en la máquina BBC con la diferencia importante que variables tales como $A% en esa máquina no se borran cuando se borra el programa de modo que se pueden pasar de un programa a otro. Esta poderosa característica o atributo no se mencionó en el manual provisional de la máquina BBC.

La tabla 3.2. da las equivalencias para ATOM de las instrucciones MICROSOFT para seccionamiento de tiras con ejemplos.

Tabla 3.2. **Instrucciones para seccionamiento de tiras. Diferencias entre MICROSOFT y ATOM**

Microsoft	Ejemplo	ACORN	Ejemplo
LEFT$(tira, número)	B$=LEFT$(A$,2)	asignar	$B=$A
		tira+número ="	$B+2=""
RIGHT$(tira,número)	B$=RIGHT$(A$,3)	tira+número ="	$B=$A+2
MID$(tira, arranque,	B$=MID$(A$,2,6)	asignar	$B=$A
final)		tira+final="	$B+6=""
		tira=tira+comienzo	$B=2

13000 Búsqueda de tiras - Primera letra dada

Esta subrutina busca en el vector de tiras aquél que tenga una primera letra determinada. El vector de tiras es B$, y el número total de elementos en el vector tiene que transferirse a la subrutina como la variable J. La variable K se usa como una señal para indicar si se ha encontrado o no una tira y se debería poner a cero antes de llamar a la subrutina. La letra elegida se asigna a la tira A$ y en 13003 se prueba para ver si cumple con la condición de carácter único. No se ha intentado restringir el uso de A$ a caracteres alfabéticos solamente; A$ puede ser numérico o incluso algunos de los signos tales como £, $, %, &, ?, *. Se puede hacer un filtrado a voluntad restringiendo el valor de ASCII de A$ a un margen particular, para indicar solamente un método.

La subrutina debería continuarse preferentemente por alguna pausa, tal como la del asterisco intermitente, para permitir que el usuario tenga tiempo para digerir los mensajes en las líneas 13009, 130012, 13014 de acuerdo con el resultado de la rutina.

Resulta también aconsejable que antes de llamar a la subrutina se use una intrucción para borrar la pantalla y evitar así confusiones visuales por exceso de información.

```
13000 REM 3.1 BUSQUEDA DE UNA TIRA—SE PRECISA DEFINIR, J,K
13001 PRINT "TECLEE LA PRIMERA LETRA, POR FAVOR";
13002 INPUT A$
13003 IF LEN (A$) > 1 THEN 13012
13004 FOR N =$ 1 TO J: REM J ES EL TOTAL DE ELEMENTOS
13005 IF A$ = LEFT$ (A$(N),1) THEN 13009
13006 NEXT N
13007 IF K < > 255 THEN 13014
13008 RETURN
13009 PRINT "ITEM";N; "ES"; B$ (N)
13010 K = 255
13011 GOTO 13006
13012 PRINT "UNA LETRA SOLO, POR FAVOR"
13013 GOTO 13001
13014 PRINT "NINGUN NOMBRE EMPIEZA CON"; A$
13015 GOTO 13008
```

Variables

PASADAS	J número de elementos
	K señalización
	B$ vector en el que se hace la búsqueda
LOCALES	N contador
	A$ letra a buscar
DEVUELTAS	B$(N) elementos seleccionados
	K señal de no encontrado (255 si no se ha encontrado)

13010 Buscador de una tira de longitud dada

Esta subrutina sirve para localizar una tira de una longitud determinada en una distribución de tiras. La longitud deseada se introduce como la variable L en la línea 13013 y se prueba para valores de cero, negativos o fraccionarios en la línea 13014. Las longitudes de las tiras de vector se comparan luego con el valor que tiene la variable L y se imprimen las cadenas con la misma longitud.

La variable de señalización K se inicializa en la línea 13014 y sirve para imprimir el mensaje de la línea 13023 si no se encuentra una tira de la longitud buscada.

El programa principal debe pasar un valor a J, el número total de elementos de vector.

```
13010 REM BUSCADOR DE TIRA DE LONGITUD DADA
13011 PRINT "DAR LA LONGITUD DE LA TIRA";
13012 K = 0
13013 INPUT L
13014 IF L < = 0 OR INT (L) < > L THEN 13025
13015 FOR N = 1 TO J
13016 IF L = LEN (A$(N)) THEN 13020
13017 NEXT N
13018 IF K = 0 THEN 13023
13019 RETURN
13020 PRINT "ITEM"; N; "ES", A$(N)
13021 K = 255
13022 GOTO 13017
13023 PRINT "NO HAY NINGUNA TIRA DE LONGITUD"; L;
      "CARACTERES".
13024 GOTO 13029
13025 PRINT "SOLO NUMEROS ENTEROS POSITIVOS, POR
      FAVOR"
13026 GOTO 13011
```

Variables

PASADAS	J número de elementos
	A$ () vector en que se hace la búsqueda
LOCALES	L Longitud de la tira
	N contador
DEVUELTAS	A$() elemento(s) seleccionado(s)
	K señal de "no encontrado" (255 si no se ha encontrado)

13020 Buscador de subtiras

Esta rutina, muy útil, busca en un vector de tiras una subtira —un conjunto de caracteres consecutivos que existen dentro de una de las cadenas de vector—

De nuevo, aquí el programa principal debe suministrar un valor para J el número total de elementos y se pide al usuario que introduzca, en la línea 13021 el grupo de caracteres A$ que hay que encontrar. Es deseable limitar la longitud de esta tira —normalmente no se quiere buscar la palabra HIPOPOTAMO dentro de otra palabra— y así se ha hecho arbitrariamente en la línea 13024 a cinco caracteres.

Este valor podría también transferirse del primer programa y tendría que igualmente insertarse en el mensaje de atención en la línea 13037.

El bucle que empieza en la línea 13025 toma las tiras del vector una por una y el bucle anidado en 13026 selecciona los puntos de arranque para grupos de letras en cada tira. La línea 13027 comprueba la igualdad entre la tira introducida A$ y un grupo de la misma longitud escogido en cada posible punto de arranque del elemento del vector seleccionado.

La variable K se usa asimismo como una señalización para indicar si se han encontrado los grupos deseados produciendo el mensaje de la línea 13035 si no se ha encontrado ninguna tira que contenga el grupo.

```
13020 PRINT "TECLEAR LA SUBTIRA A BUSCAR, POR FAVOR";
13021 INPUT A$
13022 K = 0
13023 Y = LEN (A$)
13024 IF Y > 5 THEN 13037
13025 FOR N = 1 TO J
13026 FOR X = 1 TO (LEN (B$(N))—Y+1)
13027 IF A$ = MID$ (B$(N),X,Y) THEN 13032
13028 NEXT X
13029 NEXT N
13030 IF K = 0 THEN 13035
13031 RETURN
13032 PRINT "ITEM"; N; "ES"; B$ (N)
13033 K = 255
13034 GOTO 13029
13035 PRINT "LA SUBTIRA"; A$; "NO SE HALLA EN NINGUNO DE
      LOS NOMBRES DE LA LISTA"
13036 GOTO 13031
13037 PRINT "EL MAXIMO SON CINCO LETRAS, POR FAVOR"
13038 GOTO 13020
```

Variables

PASADAS	J número de elementos
	B$ () vector de tiras a explorar

LOCALES	Y longitud de la tira a buscar tecleada
	N,X contadores
	A$ subtira que hay que encontrar
DEVUELTAS	B$ () elementos
	K señalización "NO ENCONTRADO" (255 si "no encontrado")

13030 Clasificación en columnas (números)

Esta subrutina toma un vector de números y los clasifica según el orden, de menor a mayor. Lo hace tomando el primer número y comparándolo con todos los demás, poniendo el menor primero en el vector y empezando de nuevo con el segundo elemento y así hasta que se han clasificado todos los números en el orden de menor a mayor; las instrucciones entre 13038 y 13040 cambian los números cuando sea necesario. El programa principal tiene que suministrar el número total de elementos como el valor de la variable J. Advertir el mensaje en la línea 13030.

Esta clasificación como todas las que comparan valores adyacentes es lenta cuando se tienen que clasificar muchos números. Esto es particularmente cierto cuando el vector es un vector de tiras porque el problema de obtener información inútil puede aparecer entonces; de esto hablaremos en otro momento.

```
13030 PRINT "CLASIFICACION EN PROCESO: ESPERE"
13031 FOR N = 1 TO J - 1
13032 FOR P = N + 1 TO J
13033 IF K (P) < = K (N) THEN 13038
13034 NEXT P
13035 NEXT N
13036 PRINT "CLASIFICADO..."
13037 RETURN
13038 Z = K (P)
13039 K(P) = K (N)
13040 K (N) = Z
13041 GOTO 13034
```

Variables

PASADAS	J número total de elementos en la lista
	K(J) vector de elementos
LOCALES	N,P, contadores
	Z variable para el cambio

DEVUELTAS K(J) vector clasificado
J número total de elementos

13040 Clasificación Shell-Metzner (números)

Esta es una de las clasificaciones más rápidas que se pueden escribir fácilmente en BASIC. El principio en que se basa es el de dividir el vector en secciones y comparar elementos de cada sección, intercambiando cuando sea necesario, hasta que se han puesto en orden todas las secciones. Aunque el tiempo que se necesita para clasificar unos pocos elementos no representa una ventaja sobre la clasificación por columnas (puede incluso ser más largo), la clasificación S.M. es mucho más rápida cuando hay muchos elementos.

Esta mayor velocidad no existe necesariamente cuando se clasifican tiras, a causa del problema de la obtención y liberación de memoria de trabajo.

Cada intercambio que se hace con una tira (líneas 13054 a 13056) creará una nueva tira en la memoria, y ésta se llenará normalmente con información inútil.

La rutina de recogida de memoria del ordenador entra en acción y se paraliza la rutina de clasificación y otras actividades hasta que se libera la memoria.

La mayor parte de los ordenadores no indican cuando ocurre este proceso, comparado con un trance catatónico y su efecto es prolongar la clasificación de una tira grande por un tiempo considerable.

A pesar de ello la clasificación S.M. es siempre interesante si el número de elementos excede de 50, pero la ventaja no aparecerá si se tiene que recuperar memoria. Un remedio parcial es reservar tanto espacio para la tira como permita el ordenador, otro es colocar la instrucción XX = FRE (A$) en el bucle exterior (el primer bucle FOR ... NEXT) de modo que se produzca la recogida de memoria a intervalos controlados.

Un tercer método se ofrece en la siguiente subrutina.

```
13040 PRINT "CLASIFICACION, ... ESPERE POR FAVOR"
13041 Y = 1
13042 Y = 2* Y
13043 IF Y < N THEN 13042
13044 Y = INT ((Y—1)/2)
```

```
13045 IF Y = 0 THEN 13053
13046 IT = N—Y
13047 FOR I = 1 TO IT
13048 J = I
13049 Z = J+Y
13050 IF K (Z) <= K (J) THEN 13054
13051 NEXT I
13052 GOTO 13044
13053 RETURN
13054 X = K (Z)
13055 K (Z) = K (J)
13056 K (J) = X
13057 J = J—Y
13058 IF J > 0 THEN 13049
13059 GOTO 13051
```

Variables

PASADAS	N número de elementos
	K() vector que hay que clasificar
LOCALES	Y,I,IT,J,Z, contadores
	X variable para intercambio
DEVUELTAS	K() vector clasificado

13050 Shell-Metzner con VARPTR

Muchos ordenadores usan la función VARPTR, que permite que se encuentren la dirección de la memoria y la longitud de una variable de tiras, usando las instrucciones PRINT PEEK (VARPTR (tira)).

El método estándar que se utiliza por MICROSOFT y por otras versiones de BASIC es que VARPTR (A$) devuelva una dirección numérica y la información (byte) almacenada en esta dirección es la longitud de la tira.

Las dos direcciones consecutivas próximas almacenan los bytes inferior y superior, en este orden, de la dirección en la que la cadena empieza en la memoria. Estas tres cantidades son los punteros para la tira y si estos tres punteros pueden intercambiarse, en lugar de intercambiar cada byte de dos tiras, entonces la operación de intercambio de tiras que es tan necesaria para la clasificación de las tiras se puede hacer sin generar información inútil.

Esta subrutina para clasificación de tiras usa la clasificación S.M. adaptada para el intercambio de punteros en lugar de tiras completas. Así se

elimina el problema de recolectar la información inútil y puede acelerar las clasificaciones de gran cantidad de elementos.

No hay ventajas en usar este método para un número pequeño de elementos ya que el tiempo consumido en el bucle que contiene las líneas 13061 a 13069 es alto cuando se compara con el tiempo que se necesita para un intercambio directo en el caso de un número pequeño de tiras cortas.

```
13050 PRINT "CLASIFICACION ... ESPERE POR FAVOR"
13051 Y = 1
13052 Y = 2* Y
13053 IF Y < N THEN 13052
13054 Y = INT ((Y—1)/2)
13055 IF Y = 0 THEN 13063
13056 IT = N — Y
13057 FOR I = 1 TO IT
13058 J = I
13059 Z = J+Y
13060 IF A$ (Z) < A$ (J) THEN 13064
13061 NEXT I
13062 GOTO 13054
13063 RETURN
13064 FOR L = 0 TO 2
13064 Y1 = PEEK (VARPTR(A$(Z))+L)
13066 Y2 = PEEK (VARPTR(A$(J))+L)
13067 POKE (VARPTR(A$(Z))+L),Y2
13068 POKE (VARPTR(A$(J))+L),Y1
13069 NEXT L
13070 J = J—Y
13071 IF J > 0 THEN 13059
13072 GOTO 13061
```

Variables

PASADAS	J número de elementos
	A$() vector de tira de elementos
LOCALES	Y,I,I,T,J,Z,L, contadores
	Y1,Y2 valores de pico
DEVUELTAS	A$() vector clasificado

13060 Eliminar duplicados

Esta subrutina busca en un vector de tiras los elementos repetidos y los elimina; es una función muy útil cuando hay un gran número de elementos y donde éstos se han introducido por distintos usuarios. El número total de elementos se transfiere a la subrutina asignando este valor a la variable N y los dos bucles en las líneas 13061 y 13062 comparan cada elemento por turno con todos los elementos siguientes, convirtiendo una tira repetida en espacios en blanco y reduciendo el total en uno cada vez que se encuentra un duplicado. El valor de N que se devuelve es el nuevo total de elementos únicos o no repetidos.

```
13060 PRINT "BUSQUEDA ... ESPERE POR FAVOR"
13061 FOR K = 1 TO N—1
13062 FOR J = K + 1 TO N
13063 IF A$ (K) = A$ (J) THEN 13067
13064 NEXT J
13065 NEXT K
13066 RETURN
13067 A$ (J) = A$ (N)
13068 A$ (N) = ""
13069 N = N — 1
13070 GOTO 13064
```

Variables

PASADAS	N, número de elementos
	A$() lista de tiras en la que se busca
LOCALES	K,J, contadores
DEVUELTAS	A$ lista con duplicados eliminados
	N número de elementos en la lista reducida

13070 Recogida controlada de memoria

El problema de la recolección de información inútil, al que nos hemos referido anteriormente, puede algunas veces hacerse menos obvio, realizando una recolección deliberada de información inútil en los momentos en los que el ordenador no está realizando otros trabajos; por ejemplo, después que se ha impreso un mensaje, o entre ejecuciones de clasificación. Esta subrutina que puede usarse con cualquier ordenador que tenga la instrucción FRE (tira) imprimirá un mensaje y reasignará la memoria.

La variable X puede utilizarse si se necesita; es igual al número de bytes libres del espacio no asignado a tiras.

```
13070 PRINT "SE ESTA REASIGNANDO LA MEMORIA: ESPERE"
13071 X = FRE (ZZ$)
```

Variables

PASADAS	ninguna
LOCALES	ZZ$ tira simulada, no utilizada
DEVUELTAS	X número de bytes libres para almacenamiento de tiras

13080 Clasificación rápida de números

Esta subrutina más bien larga, es una de las más rápidas conocidas para clasificar números pero tiene la desventaja de utilizar muchas variables y de ocupar un gran espacio de memoria.

Para una máquina de 16 K (o más), con posibilidad de usar líneas con varias instrucciones y con un gran número de nombres de variables permitidas, es la preferida para la clasificación numérica de grandes vectores o tablas.

Como todas las rutinas de clasificación, se basa en el principio de descomponer el vector en grupos y comparar elementos de distintos grupos, pero el modo de seleccionar los grupos y de comparar los números es más complicado que el que usa la rutina Shell Metzner.

A causa de la obtención de memoria no hay ninguna ventaja en usar esta rutina para tiras, a menos que se disponga de una máquina que haya declarado las longitudes de las tiras y que no produzca información inútil. Hasta donde yo sé la rutina funciona adecuadamente con tiras en el micro BBC.

```
13080 PRINT "CLASIFICACION ... POR FAVOR ESPERE"
13081 P = 1
13082 Q (1) = 1
```

```
13083 R (1) = N
13084 U = Q (P)
13085 D = R (P)
13086 P = P — 1
13087 J = U
13088 L = D
13089 X = K (INT ((U+D)/2))
13090 IF K (J) < X THEN 13107
13091 IF X < K (L) THEN 13109
13092 IF J > L THEN 13098
13093 W = K (J)
13094 K (J) = K(L)
13095 K (L) = W
13096 J = J + 1
13097 L = L — 1
13098 IF J <= L THEN 13090
13099 IF J >= D THEN 13103
13100 P = P + 1
13101 Q (P) = J
13102 R (P) = D
13103 D = L
13104 IF U < D THEN 13087
13105 IF P <> 0 THEN 13084
13106 RETURN
13107 J = J + 1
13108 GOTO 13090
13109 L = L — 1
13110 GOTO 13091
```

Variables

PASADAS	N, número de elementos
	K() vector que hay que clasificar
LOCALES	P, contador
	Q() R() vector de contadores
	U,D,J,L,X,W, almacenamiento temporal
DEVUELTAS	K() vector clasificado

13090 Clasificación de sub-vectores

Esta subrutina permite separar un grupo de elementos de un vector de tiras para formar otro vector. Esto puede ofrecer varias ventajas entre las

que no es la menor que se pueda clasificar el vector obtenido rápidamente si se requiere.

La característica elegida para los elementos en este ejemplo ha sido la letra inicial, pero cualquier otra característica (longitud de la tira, tener un determinado grupo de letras o caracteres etc.) puede elegirse, usando las subrutinas que se han descrito anteriormente.

La variable A$ se usa como el identificador para el vector B$ en el que el número total de elementos N debe traspasarse a la rutina. Los elementos seleccionados se ponen en una nueva distribución C$ cuyo número total es Q. Se imprimirá un mensaje en el caso de que no se encuentren tiras del grupo requerido.

```
13090 PRINT "POR FAVOR, TECLEE LA LETRA INICIAL A
      SELECCIONAR
13091 K = 0
13092 Q = 0
13093 INPUT A$
13094 IF LEN (A$) > 1 OR LEN (A$) < 1 THEN 13100
13095 FOR J = 1 TO N
13096 IF A$ = LEFT$ (B$(J),1) THEN 13104
13097 NEXT J
13098 IF K = 0 THEN 13102
13099 RETURN
13100 PRINT "SOLO UNA LETRA, POR FAVOR"
13101 GOTO 13090
13102 PRINT "NINGUN NOMBRE COMIENZA CON"; A$
13103 GOTO 13099
13104 K = 225
13105 Q = Q + 1
13106 C$ (Q) = B$ (J)
13107 GOTO 13097
```

Variables

PASADAS	N, número de elementos en el vector principal
	B$ vector de tiras
LOCALES	J, contador
DEVUELTAS	K, identificación "no encontrado" (255 si no se encuentra)
	Q, número de elementos encontrados.
	C$() vector de elementos encontrados.

13100 Selección de un elemento

La instrucción READ... DATA que existe en casi todos los microordenadores (con las únicas excepciones de los ZX80/81 y ACORN ATOM) es útil para leer los elementos de una distribución, pero está limitada al ser estrictamente secuencial.

Esta subrutina permite elegir un elemento de una lista DATA (de datos) especificando un número que se asigna a la variable K. Se leen entonces K elementos de la lista de datos (DATA) y asignados a su vez a la variable A$ (se supone un conjunto de datos de tiras ya que cualquier dato puede leerse como una tira).

El valor de A$ que se devuelve al programa principal será entonces el elemento correspondiente al número que se introdujo. La instrucción RESTORE en la línea 1301 es necesaria para asegurarse que la lista de datos se lee desde el principio cada vez que se usa la rutina.

El número total de elementos N se debe traspasar a la rutina para evitar que aparezca un mensaje de error (OUT OF DATA) si el valor seleccionado es demasiado alto.

```
13100 PRINT "QUE ELEMENTO NUMERICO SE REQUIERE"
13101 RESTORE
13102 INPUT K
13103 IF K <=0 OR     K > N     THEN  13108
13104 FOR J = 1 TO K
13105 READ A$
13106 NEXT J
13107 RETURN
13108 PRINT "NUMEROS ENTRE 1 Y";N;" SOLAMENTE
      POR FAVOR"
13109 GOTO 13100
```

Variables

PASADAS	N numero máximo de elementos en la lista de datos
	A$ elementos de los datos en la lista de caracteres
LOCALES	J contador
DEVUELTAS	K número del elemento
	A$ elemento seleccionado

13110 Búsqueda de un elemento en una lista ordenada

La búsqueda de un elemento determinado de una lista en las que sus elementos están distribuídos al azar, requiere la comprobación de cada uno de ellos, pero si la lista se ha clasificado previamente, entonces se realiza otro tipo de búsqueda; se puede usar la búsqueda binaria.

La diferencia es la misma que resulta de comparar el tiempo necesario para encontrar una palabra en un libro mirando las páginas secuencialmente o mirando el índice.

El principio usado es dividir la lista en dos secciones y decidir en cuál de ellas se encontrará el elemento buscado.

Dicha sección se divide a su vez en otras dos, y la parte que contiene el elemento se selecciona; se repite el procedimiento hasta que se encuentra el elemento. La gran ventaja es que se necesitan comparativamente pocas comprobaciones incluso si la lista es muy larga, y los elementos pueden encontrarse en un tiempo relativamente corto.

Cuando hay que buscar elementos de una lista dada, vale la pena clasificar la lista primero, grabar la lista clasificada y utilizar esta rutina posteriormente.

El número total de elementos se traspasa como un valor para N, el vector en el que se realiza la búsqueda es B$ y el elemento buscado A$.

```
13110 PRINT "TECLEE EL NOMBRE BUSCADO",
13111 INPUT A$
13112 J = N + 1: REM N ES TOTAL
13113 N1 = J
13114 N2 = INT ((N1+1)/2)
13115 IF N2 = 0 OR N2 > N THEN 13123
13116 IF A$ < B$ (N2) THEN 13121
13117 IF A$ = B$ (N2) THEN 13124
13118 N1 = N2+J
13119 N3 = N 2
13120 GOTO 13124
13121 N1 = N2+N3
13122 GOTO 13114
13123 PRINT A$; "NO ESTA EN LA LISTA"
13124 RETURN
```

Variables

PASADAS	N, número total de elementos en la lista
	B$() vector de elementos
LOCALES	J,N1,N2,N3 contadores
DEVUELTAS	N2, número del elemento encontrado

4

Operaciones con vectores y matrices

Algunos de los primeros textos en Basic que se ofrecían al usuario de microordenadores, contenían referencias a instrucciones tales como MATPRINT que nunca han existido en los microordenadores que yo sepa, revelando que los libros de hecho, se habían escrito para los usuarios de ordenadores de mayor tamaño.

En general, todos los ordenadores permiten el uso de vectores que tienen un subíndice de la forma N(1) N(2) etc.

La mayor parte de los micros también permite la formación de una variable matricial que es simplemente una matriz de dos dimensiones tales como N(1,1), N(1,2) etc.

Si se usan otras instrucciones más avanzadas es debido al tamaño de la ROM y a la importancia que el diseñador ha dado a las rutinas con matrices en comparación con otros aspectos, como color de los gráficos etc.

Las subrutinas en esta sección se han diseñado para que se puedan hacer operaciones útiles con vectores y matrices en máquinas que tienen solamente un mínimo de capacidades en este campo.

Las últimas rutinas del capítulo son probablemente más interesantes al lector que se dedique a aplicaciones científicas y matemáticas más bien que a aplicaciones comerciales.

Empecemos, sin embargo, con una breve descripción de la obtención de números aleatorios y su sintaxis en algunas máquinas populares.

RND

La instrucción RND se usa generalmente en micros para obtener números aleatorios, pero la sintaxis varía ligeramente de un ordenador a otro. El sistema más lógico es el usado en el micro BBC en el que la instrucción RND (6) por ejemplo producirá un número seleccionado aleatoriamente en el rango del 1 al 6 ambos inclusive. Este ejemplo da el número aleatorio correcto, del mismo modo que el que se obtendría con un dado.

La tabla 4.1 indica algunas variaciones de sintaxis en ésta y otras máquinas.

Tabla 4.1. **Sintaxis de RND**

1. **Máquina BBC** (basado en la guía provisional del usuario)
   RND usada sola genera un número elegido aleatoriamente entre —2147483648 y + 2147483647.
   RND (—N) imprime el valor de N y resiembra el generador de números aleatorios con un valor que está basado en N.
   RND (0) repite el último número aleatorio generado por RND (1).
   RND (1) genera un número escogido aleatoriamente entre 0 y 0.999999.
   RND (X) genera un número aleatorio entre, o puede incluir 1 y el valor de X.
2. **ZX81**
   RND se usa siempre sola y devuelve un número entre 1 y 65536 inclusive.
   La siembra del generador de números aleatorios se altera con RAND X donde X es el número sembrado. Un valor dado de X siempre generará la misma secuencia de números aleatorios.
3. **ACORN ATOM**
   RND devuelve un número aleatorio que está entre —2147483648 y +2147483647.
   Restaura la siembra insertando un nuevo número entre las direcciones 8 a 12 inclusive. Para números aleatorios entre A y B, donde A y B son números enteros, usar ABSRND%(B—A)+A
4. **TRS 80/Video Genie**
   RND (0) devuelve con un dígito de simple precisión (6 dígitos) entre 0 y 1.
   RND(N) donde N es un número entero devolverá un número aleatorio entre 1 y N. La instrucción RANDOM se usa para resembrar el generador.
5. **CMB PET 2000**
   RND (—N) produce el mismo número en cada llamada.
   RND (0) produce la misma secuencia de números aleatorios.
   RND (+N) produce una secuencia distinta en cada llamada.
   Cada número aleatorio está entre 0 y 1.

6. APPLE 2

RND (—N) produce un número aleatorio entre 0 y —N+1
RND (+N) produce un número aleatorio entre 0 y N—1

El principal problema de RND es el de producir un número aleatorio verdadero, porque no hay proceso operativo en el microprocesador que sea verdaderamente aleatorio; todas las señales y números se obtienen por medio de algún tiempo de cálculo o proceso lógico.

Algunas máquinas basadas en el Z-80 producen aleatoriedad basando sus números aleatorios en el contenido del registro de refresco de dicho microprocesador, descansando en el hecho que el contenido del registrador R tiene menos relación a los procesos que se realizan por el ordenador que las de otro registrador.

El problema no es simple; es relativamente fácil generar números que son bastante aleatorios para algunos juegos, pero producir números que puedan ser clasificados o definidos como aleatorios para muestras estadísticas es otra cosa. Antes de usar RND para análisis serios de estadísticas, se deberían comprobar los números obtenidos para verificar si son realmente aleatorios o por el contrario siguen una distribución no aleatoria.

Un método muy común de generar una serie de números aleatorios es empezar con un valor "sembrado" y generar otros números a partir de el.

Dos fórmulas ligeramente distintas son:

a) r' = (r * a) MOD B
donde r es el número aleatorio previo
r' es un nuevo número aleatorio
B es un número primo
a es una "semilla" que si está bien elegida producirá números aleatorios con esta fórmula

b) r' = (a * r + c) MOD D
Como antes pero D es un número grande y no necesita ser primo y a y c son seleccionados.
Experimentos realizados sugieren que con D = 65536 y c un número impar, a puede ser 293,389, ó 1509 con buenos resultados.

El Z-80 y Z-81 usan B = 65537 y a = 75 en su forma de expresión (a). Una vez que se ha producido un número aleatorio fraccionario se puede manipular por multiplicación y redondeamiento para obtener números

enteros dentro del margen que se necesite. El uso de RAMDOMISE (RAMDOMIZE) en muchos diseños de ordenador permite que se pueda cambiar el valor sembrado para evitar posible repetición de secuencias.

Para más información sobre números aleatorios ver la obra de Knuth "Art of Computer Programming".

14000 Matriz de Pares

Esta subrutina simple permite que pares de elementos se introduzcan en una matriz de tiras A$. El número de pares debe especificarse en la variable J.

La elección de pares hace que ésta sea particularmente útil para asociar preguntas con respuestas y si el ordenador puede clasificar matrices así como distribuciones de tiras, permite clasificar juntos pares de elementos, mucho más simplemente que asignando dos variables de tiras separadas e intercambiándolas.

```
14000 PRINT "POR FAVOR, TECLEAR"; J; "PAREJAS DE
      ELEMENTOS"
14001 FOR N = 1 TO J
14002 INPUT A$ (N,1), A$ (N,2)
14003 NEXT N
14004 RETURN
```

Variables

PASADAS	J, número de parejas
LOCALES	N, contador
DEVUELTAS	A$() matriz de elementos

14010 Formación de una matriz N × M

Esta subrutina permite poner a los elementos en una distribución bi dimensional. Los números de las filas/columnas se imprimen antes de que se introduzca cada elemento (línea 14013) de modo que el usuario puede ver los números que se han asignado.

```
14010 PRINT "POR FAVOR, TECLEAR"; J; "CONJUNTOS DE"; P;
      "ELEMENTOS"
14011 FOR N = 1 TO J
14012 FOR M = 1 TO P
14013 PRINT "ELEMENTO" N; CHR$ (8) "." M;
14014 INPUT A$ (N,M): REM RECUERDE DIMENSIONAR A$
14015 NEXT M
14016 NEXT N
14017 RETURN
```

Variables

PASADAS	J, número de conjuntos
	P, número de elementos en cada conjunto
LOCALES	N,M, contadores
DEVUELTAS	A$(,) matriz de elementos

14020 Impresión de dos columnas

Los elementos que se han introducido en una matriz de dos columnas por la subrutina 14000 pueden imprimirse en formato de dos columnas por esta subrutina.

El número total de filas tiene que traspasarse como el valor de J.

```
14020 PRINT "RESULTADOS"
14021 FOR N = 1 TO J
14022 PRINT A$ (N,1); TAB (20)A$(N,2)
14023 NEXT N
14024 RETURN
```

Variables

PASADAS	J, número de filas
	A$(,) matriz de elementos
LOCALES	N, contador
DEVUELTAS	ninguna

14030 Impresión de una matriz N × M

Pocos ordenadores (por no decir ninguno) poseen la instrucción MATPRINT de modo que una matriz bidimensional tiene que imprimirse en filas y columnas por una rutina de este tipo. Se debe tener cuidado de los números TAB en la línea 14033, porque la matriz puede tener más columnas de las que pueden imprimirse en la pantalla y no hay ninguna forma fácil para eludir este problema.

* A partir de aquí en este capítulo, las subrutinas son de más interés para usuarios matemáticos o científicos. Se han descrito en menor detalle porque los futuros usuarios conocerán generalmente lo que significa la función de cada subrutina *.

```
14030 PRINT "RESULTADOS"
14031 FOR N = 1 TO J
14032 FOR M = 1 TO P
14033 PRINT TAB (10*(M—1)) A$ (N,M);
14034 NEXT M
14035 PRINT
14036 NEXT N
14037 RETURN
```

Variables

PASADAS	J,P, totales de filas y columnas
	A$(,) matriz de elementos
LOCALES	N,M contadores
DEVUELTAS	ninguno

14040 Multiplicación escalar — Matriz de tres elementos

La multiplicación escalar de una matriz significa multiplicar cada elemento por el mismo valor y para una matriz tridimensional esto significa usar tres bucles FOR ... NEXT.

Los valores máximos de cada dimensión deben traspasarse como las variables N1, N2 y N3 y el factor multiplicador es la variable Z. El resultado es la matriz B (I,J,K,).

```
14040 FOR I = 1 TO N1
14041 FOR J = 1 TO N2
14042 FOR K = 1 TO N3
14043 B(K,J,I) = A(K,J,I)* Z
14044 NEXT K
14045 NEXT J
14046 NEXT I
14047 RETURN
```

Variables

PASADAS	N1,N2,N3, números dimensionales de la matriz
	Z número multiplicador
	A (,) matriz de números
LOCALES	I,J,K, contadores
DEVUELTAS	B (,) matriz de números

14050 Transposición de una matriz (dos dimensiones)

Esta rutina transpone (intercambia filas y columnas) una matriz de dos dimensiones A (I,J,) en la nueva matriz B(J,I). Las dimensiones N1 y N2 deben traspasarse desde el programa principal.

```
14050 FOR I = 1 TO N1
14051 FOR J = 1 TO N2
14052 B (J,I) = A (I,J)
14053 NEXT J
14054 NEXT I
14055 RETURN
```

Variables

PASADAS	N1,N2 dimensiones de la matriz
	A(,) matriz
LOCALES	I,J, contadores
DEVUELTAS	B(,) matriz resultante

14060 Adición de matrices — caso Bidimensional

Esta subrutina suma dos matrices bidimensionales A y B para formar la nueva matriz C.

```
14060 FOR J = 1 TO N2
14061 FOR I = 1 TO N1
14062 C (I,J) = A (I,J)+B (I,J)
14063 NEXT I
14064 NEXT J
14065 RETURN
```

Variables

PASADAS	N1,N2 dimensiones de la matriz
	A(,) B(,) matrices
LOCALES	I,J, contadores
DEVUELTAS	C(,) matriz resultante

14070 Multiplicación de elementos

Las matrices A y B se multiplican elemento a elemento para formar los elementos de la nueva matriz C.

```
14070 FOR J = 1 TO N2
14071 FOR I = 1 TO N1
14072 C (I,J) = A(I,J)* B (I,J)
14073 NEXT I
14074 NEXT J
14075 RETURN
```

Variables

PASADAS	N1,N2 dimensiones de la matriz
	A(,) B(,) matrices a multiplicar
LOCALES	I,J, contadores
DEVUELTAS	C(,) matriz producto

14080 Multiplicación de matrices

La matriz C se forma multiplicando la matriz A por la matriz B. Advertir que hay que traspasar a esta subrutina tres números dimensionales.

```
14080 FOR I = 1 TO N1
14081 FOR J = 1 TO N2
14082 C(I,J) = 0
14083 FOR K = 1 TO N3
14084 C (I,J) = C(I,J)+A(I,K)* B(K,J)
14085 NEXT K
14086 NEXT J
14087 NEXT I
14088 RETURN
```

Variables

PASADAS	N1,N2,N3 dimensiones de las matrices: A(N1 × N3), B(N3 × N2), C(N1 × N2) A(,) B(,) matrices
LOCALES	I,J,K, contadores
DEVUELTAS	C (,) matriz producto

Las próximas dos subrutinas son largas, pero extremadamente útiles. Para comprobar que se han introducido correctamente, el método más simple es escribir un corto programa de llamada que alimenta un juego de valores cuyo resultado puede comprobarse fácilmente.

14090 Inversión de una matriz — Bidimensional

El mensaje en la primera línea es necesario por el tiempo que puede durar esta rutina. La variable N es el número de dimensiones (número de filas igual al de columnas) de la matriz V que se usa para la entrada a la rutina.

La rutina invertirá entonces la matriz si esto es posible y transfiere el resultado como una matriz K. Si no hay solución se imprime un mensaje con dicha información.

Esto también pone la variable XX a 255 de modo que cualquier impresión que seguiría normalmente para visualizar la matriz puede suprimirse.

```
14090 PRINT "ESPERE POR FAVOR, ESTOY CALCULANDO"
14091 XX = 0
14092 FOR J = 1 TO N
14093 FOR L = 1 TO N
14094 K (J,L) = 1 — ABS (SGN (J—L))
14095 NEXT L
14096 NEXT J
14097 FOR J = 1 TO N — 1
14098 IF V (J,J) = 0 THEN 14115
14099 FOR L = J + 1 TO N
14100 M = V (L,J)/V(J,J)
14101 FOR P = 1 TO N
14102 V (L,P) = V (L,P) — M*V (J,P)
14103 K (L,P) = K (L,P) — M*K (J,P)
14104 NEXT P
14105 NEXT L
14106 NEXT J
14107 FOR J = N TO 1 STEP — 1
14108 FOR P = 1 TO N
14109 FOR L = J + 1 TO N
14110 K (J,P) = K (J,P) — V (J,L)* K (L,P)
14111 NEXT L
14112 K (J,P) = K (J,P)/V(J,J)
14113 NEXT P
14114 NEXT J
14115 RETURN
14116 PRINT "NO TIENE SOLUCION"
14117 XX = 255
14118 GOTO 14115
```

Variables

PASADAS	N, número de filas = número columnas
	V (,) matriz a invertir
LOCALES	XX,J,L,P, números intermedios
DEVUELTAS	K (,) matriz invertida

14100 Solución de ecuaciones simultáneas

La solución de ecuaciones simultáneas puede seguir el proceso de la inversión de una matriz, pero esta subrutina realiza todo el proceso. Los

coeficientes de las incógnitas se introducen en orden (x,y,z, por ejemplo) como elementos de la matriz K y los números como la matriz G. Las soluciones se imprimen en el orden x y z de las incógnitas como la matriz L. Por ejemplo:

$$\begin{array}{lll} 3x + 2y = 12 & 3 & 2 \\ 4x + y = 11 & 4 & 1 \end{array} = \begin{array}{l} 12 \\ 11 \end{array}$$

Asignación de matriz

K (1,1) = 3 K (1,2) = 2
K (2,1) = 4 K (2,2) = 1

Asignación de vectores

G (1) = 12
G (2) = 11

Llamar la subrutina
Imprimir el vector L

L (1) = 2
L (2) = 3

Por eso x = 2, y = 3. Son las soluciones

```
14100 PRINT "ESPERE POR FAVOR, ESTOY CALCULANDO
14101 XX = 0
14102 FOR J = 1 TO N — 1
14103 Y = J
14104 X = K (J,J)
14105 FOR H = J + 1 TO N
14106 IF K (H,J) <= X THEN 14109
14107 Y = H
14108 X = K (J,H)
14109 NEXT H
14110 IF X = 0 THEN 14137
14111 IF Y = J THEN 14120
14112 FOR F = 1 TO N
14113 Z = K (J,F)
14114 K (J,F) = K (Y,F)
14115 K (Y,F) = Z
14116 NEXT F
14117 Z = G (J)
14118 G (J) = G(Y)
14119 G (Y) = Z
14120 FOR H = J + 1 TO N
14121 X = K (H,J)/K(J,J)
14122 FOR F = J TO N
```

```
14123 K (H,F) = K (H,F) — X*K (J,F)
14124 NEXT F
14125 G (H) = G (H) — X*G (J)
14126 NEXT H
14127 NEXT J
14128 L (N) = G(N)/ K (N,N)
14129 FOR J = N — 1 TO 1 STEP — 1
14130 Z =0
14131 FOR H = J + 1 TO N
14132 Z = Z + L (H)* K (J,H)
14133 NEXT H
14134 L (J) = (G(J)—Z)/ K (J,J)
14135 NEXT J
14136 RETURN
14137 PRINT "NO TIENE SOLUCION"
14138 XX = 255
14139 GOTO 14136
```

Variables

PASADAS	N, número de ecuaciones
	K, matriz de incógnitas
	G, vector de números
LOCALES	XX,J,Y,H, F contadores
	X, Z almacenamiento temporal de números
DEVUELTAS	L (,) vector de resultados

5

Gráficos y diagramas

Los diseñadores de ordenadores tienen sus preferencias respecto a la parte del sistema operativo del ordenador que se dedicará a la tarea de obtener representaciones gráficas. Mi opinión es que bastaría la capacidad de imprimir un punto en cualquier lugar que se desee de la pantalla, porque no considero que el dibujo de gráficos sea una característica que se necesite o se use con frecuencia.

La capacidad normal para dibujar gráficos denominada de baja resolución (que curiosamente algunos fabricantes denominan de alta resolución) es la que consiste en poseer un conjunto de números codificados, correspondiéndose cada uno con una forma gráfica determinada, y que se imprimirá en la pantalla respondiendo a la instrucción PRINT CHR$ (n), (n es el número), pero como cada diseñador utiliza un juego diferente de números codificados y un juego también distinto de gráficos, no hay manera de indicar cómo se puede obtener una forma o estructura gráfica con todos los ordenadores (4).

Este capítulo indica por lo aducido anteriormente, solamente unos pocos métodos generales gráficos que casi todos los ordenadores pueden usar, en vez de rutinas detalladas que sólo servirían para un ordenador determinado.

El tema de dibujar diagramas en la pantalla o en papel usando una impresora se trata con más detalle, ya que es una forma importante de visualizar información para usos científicos, y no necesita recurrir a ningún tipo de símbolo gráfico.

---

(4) En el IBM PC existen también los caracteres o dibujos formados por trozos de rectángulos (tipo TRS-80), si bien se han reducido en favor de otros símbolos (si el programador lo desea se pueden reconstruir todas las combinaciones de los rectángulos) (N. del Ed.)

SET/RESET

SET y RESET son las instrucciones usadas en un tipo de sistema gráfico de baja resolución en los ordenadores que trabajan con Microsoft Basic.

La pantalla se divide en unidades elementales llamadas *pixels* (derivada de las palabras inglesas picture element) cada espacio impresor contiene seis, dos a lo ancho y tres a lo largo. Por cada columna de texto hay dos columnas de pixels, y por cada fila de texto hay tres filas de pixels.

El TRS 80 con una pantalla que contiene 64 columnas por 16 filas, tiene por lo tanto una capacidad gráfica de 128 por 48 pixels, y estos números pueden usarse para seleccionar un pixel determinado. Un sistema similar se usa también para los gráficos Teletext-mode de la máquina BBC.

Antes de usar la instrucción SET se debe encontrar en la pantalla la posición de los pixels, y esto se logra fácilmente usando un mapa en video de la pantalla, que existe en casi todas las máquinas, que muestra los números de las columnas y de las filas de los pixels. La instrucción SET (X,Y) donde X es el número de referencia de la columna y el número de referencia de la fila hará que se ilumine la posición o pixel correspondiente a dichas coordenadas.

Por ejemplo SET (64,2) iluminará el pixel central en la segunda línea gráfica (el centro de la primera línea del texto) en el TRS 80. La instrucción SET se puede usar en cualquier lugar del programa, y los gráficos resultantes se pueden mezclar con el texto, con tal de que se tengan en cuenta los diferentes tamaños de los números de referencia.

RESET es la instrucción correspondiente, RESET (X,Y) hará que el elemento cuya posición viene determinada por las coordenadas, adquiera el color del fondo de la pantalla (normalmente netro). Si el pixel no estaba iluminado no ocurrirá nada.

Usando SET y RESET alternativamente se puede hacer intermitente un pixel u obtener un efecto de movimiento lento, al iluminar un pixel en un lado de un bloque, y extinguiéndolo en el otro lado del bloque.

La instrucción POINT es útil para usar en conjunción con SET y RESET. Se usa con las instrucciones SET/RESET del TRS 80 principalmente para interrogar las diversas posiciones de la pantalla para descubrir qué pixels están iluminados. POINT está seguido de las coordenadas del punto entre paréntesis. Es una instrucción de búsqueda que en la máquina TRS 80 devolverá un —1 si el pixel está iluminado y un 0 si por el contrario no lo está.

Se puede por lo tanto usar POINT para evitar escribir sobre un cuadro determinado o para detectar límites en dibujos complicados, tales como las

paredes de un laberinto, etc. Su acción es sin embargo muy lenta para poder usarse con juegos animados.

Se puede usar POINT en un bucle para buscar todos los pixels que estén iluminados, y con SET, RESET y RND, puede añadir o eliminar pixels produciendo unos efectos similares a los del programa denominado Juego de Vida.

La instrucción POINT en la máquina BBC devuelve el número del código del color de la posición de la pantalla interrogada. La instrucción tiene dos coordenadas en el orden X Y y el número obtenido dependerá de la modalidad del color que se utiliza, aunque la acción no sirve para MODE 7 (Teletext mode).

15000 Imprimir un punto

Esta subrutina es simple pero fundamental. Dado un número asignado a la variable Y, esta rutina imprime un punto en esta posición tabulada y después retorna. Por supuesto que se pueden usar otros caracteres en lugar del punto y puede usarse con un bucle para imprimir una fila de caracteres.

```
15000 PRINT TAB (Y) "."
15001 REM SE PUEDE USAR *, O ALGUNA BARRA
15002 RETURN
```

Variables

PASADAS	Y posición tabulada
LOCALES	ninguna
DEVUELTAS	ninguna

15010 Carácter desplazable (versión 1)

Esta subrutina usa la instrucción PRINT@ que existe en la mayor ROM de muchos ordenadores actuales. El efecto de la rutina es el de imprimir un asterisco o cualquier otro símbolo seleccionado, en una posición en la pantalla determinada por el valor de la variable Y, y borrar el

espacio que la precede. Usando la subrutina con un bucle se puede obtener un carácter que se mueve.

```
15010 PRINT@ Y, "*";
15011 PRINT@ (Y—1), "";
15012 RETURN
```

Variables

PASADAS	Y posición de PRINT
LOCALES	ninguna
DEVUELTAS	ninguna

15020 Carácter movible (versión 2)

Con las máquinas que no disponen de la instrucción PRINT@ esta versión del carácter movible usa TAB en conjunción con el código de control pàra desplazar el cursor hacia arriba.

Parecería más simple usar PRINTTAB (Y—1) ""; seguido por PRINTTAB (Y) "*", pero esto duplica el espacio del asterisco en muchas máquinas.

```
15020 PRINT TAB (Y—1) ""
15021 PRINT CHR$ (27);
15022 PRINT TAB (Y) "*"
15023 PRINT CHR$ (27);
15024 RETURN
```

Variables

PASADAS	Y posición de tabulación
LOCALES	ninguna
DEVUELTAS	ninguna

15030 Subrayado (versión 1)

Esta rutina utiliza un guión para subrayar pero se podría obtener una línea más sólida con un bloque gráfico. La variable X se habrá obtenido del programa principal o desde una rutina usada para conseguir la posición de tabulación para centrar el título en la pantalla.

```
15030 PRINT TAB (X);
15031 FOR XX = 1 TO LEN (A$)
15032 REM A$ ES EL TITULO DE CABECERA
15033 PRINT TAB (XX) "—";
15034 NEXT XX
15035 RETURN
```

Variables

PASADAS	X posición de tabulación
	A$ Título
LOCALES	XX para el subrayado
DEVUELTAS	ninguna

15040 Subrayado (versión 2)

La instrucción STRING$ no existe en algunas de las máquinas más pequeñas, pero es muy útil para crear tiras de caracteres. La rutina usa la instrucción para producir una tira de signos "igual" que servirá para subrayar un título. El formato de la instrucción permite usar bloques gráficos; por ejemplo STRING$ (20,162) producirá una tira de 20 bloques gráficos cuyo número de código es 162.

```
15040 XX = LEN (A$)
15041 PRINT TAB (X) STRING$ (XX, "=")
15042 RETURN
```

Variables

PASADAS	X, número de posición
	A$ título
LOCALES	XX, número de subrayado
DEVUELTAS	ninguna

15050 Subrayado (versión 3)

Para los ordenadores que no tienen la instrucción STRING$ esta subrutina permite reunir una tira de caracteres en una variable de tira que puede imprimirse con una instrucción.

La ventaja es que la acción de formar la tira se separa de su impresión, permitiendo que la instrucción PRINT opere mucho más rápidamente que sería posible si fuera usado en un bucle.

```
15050 ZZ$ = ""
15051 FOR K = 1 TO LEN (A$)
15052 ZZ$ = ZZ$ + "="
15053 NEXT K
15054 PRINT TAB (X) ZZ$
15055 RETURN
```

Variables

PASADAS	X, posición tabulada para el título
	A$ título
LOCALES	K contador
	ZZ$ tira subrayada
DEVUELTAS	ninguna

15060 Sobrerrayado. Subrayado (versión 1)

Esta rutina imprime una fila completa de asteriscos, un título, luego otra fila de asteriscos.

Ver también la subrutina 11050.

```
15060 FOR K = 1 TO Z
15061 REM Z CARACTERES POR LINEA
15062 PRINT TAB (K) "*";
15063 NEXT K
15064 PRINT TAB (X) A$
15065 FOR K = 1 TO Z
15066 PRINT TAB (K) "*";
15062 NEXT K
15068 RETURN
```

Variables

PASADAS	Z caracteres por línea
	X posición de impresión del título
	A$ título
LOCALES	K contador
DEVUELTAS	ninguno

15070 Sobrerrayado, Subrayado (versión 2)

Esta es la misma rutina que la 15060 con el bucle puesto en la forma de una subrutina.

```
15070 GOSUB 15074
15071 PRINT TAB (X) A$
15072 GOSUB 15074
15073 RETURN
15074 FOR K = 1 TO Z
15075 PRINT TAB (K) "*";
15076 NEXT K
15077 RETURN
```

Variables

PASADAS	Z, caracteres por línea
	X, posición de impresión del título
	A$ título

LOCALES	K contador
DEVUELTAS	ninguna

15080 Reunión de Gráficos

Los caracteres gráficos pueden imprimirse uno a la vez por instrucciones del tipo PRINT CHR$ (166) usando los números que están asignados a formas gráficas en su propio ordenador. Esto es adecuado cuando únicamente se imprimen formas simples, pero cuando se reúnen varias para formar una figura entonces se puede usar una subrutina de este tipo.

Los valores G1 a G5 son números de código de gráficos que deben suministrarse por el programa principal junto con el número de tabulación Y.

La ventaja de esta simple rutina es que varias formas distintas pueden imprimirse usando la misma subrutina simplemente pasando valores apropiados de G1-G5 incluyendo espacios en blanco si es necesario.

```
15080 PRINT TAB (Y) CHR$ (G1); CHR$ (G,2); CHR$ (G3)
15081 PRINT TAB (Y+1) CHR$ (G4); CHR$ (G5)
15082 RETURN
```

Variables

PASADAS	Y número de tabulación
	G1,G2,G3,G4,G5, números de códigos de gráficos
LOCALES	ninguna
DEVUELTAS	ninguna

15090 Cadenas de formas gráficas

Un desarrollo del método demostrado en 15080 es la reunión de un número de caracteres gráficos en una única tira, que puede entonces imprimirse. Esta rutina reúne caracteres gráficos en tres tiras que se pueden imprimir separadamente o reunidas. Tratando la forma total como una única tira, la animación se hace más fácil y se mejora mucho la velocidad de impresión.

```
15090 G1$ = CHR$ (130) + CHR$ (132) + CHR$ (146) CHR$ (11)
15091 G2$ = CHR$ (133) + CHR$ (166)
15092 G3$ = CHR$ (184)
15093 PRINT TAB (Y) G1$
15094 PRINT TAB (Y+1) G2$
15095 PRINT TAB (Y) G3$
15096 RETURN
```

Variables

PASADAS	Y número de tabulación
LOCALES	G1$, G2$, G3$ tira de gráficos
DEVUELTAS	ninguna

15100 Dibujar Gráficos

Algunas máquinas tienen la posibilidad de dibujar y borrar gráficos (SET, RESET). La instrucción SET viene seguida por los números X e Y entre paréntesis que determinan la posición de un pequeño elemento rectangular de la pantalla llamada pixel. El resultado es iluminar dicho elemento rectangular con coordenadas X e Y. La instrucción RESET (X,Y) por el contrario, extinguirá la iluminación de dicho elemento.

Otras máquinas tienen la instrucción POS (X,Y) que devolverá un 1 si el elemento está iluminado y 0 si no lo está (para detectar límites dibujados en la pantalla). SET y RESET se pueden usar también para lograr formas que no se pueden hacer convenientemente con los bloques gráficos estándar.

El principal problema que se encuentra con el uso de SET y RESET es que estas instrucciones operan sobre sólo un elemento (pixel) a la vez, de modo que formar un dibujo o modelo grande es un proceso muy largo y tedioso. El método más fácil es usar la rutina conjuntamente con la instrucción READ ... DATA, introduciendo pares de coordenadas X e Y y utilizando cada par con una instrucción SET o RESET.

```
15100 FOR N = 1 TO K
15101 READ X,Y
15102 SET (X,Y)
15103 NEXT N
15104 RESTORE
15105 RETURN
```

Variables

PASADAS	K número de pixels
	X, Y pares de datos
LOCALES	N, contador
DEVUELTAS	ninguna

15110 Introducción de gráficos en la memoria

El método gráfico más rápido es introducir el carácter del código gráfico directamente en la sección de memoria usada por la pantalla (sólo es posible si el ordenador tiene un mapa de memoria de la pantalla). La instrucción POKE viene seguida por una dirección numérica, una coma y el código numérico del gráfico.

En el ejemplo se ilustra el uso de POKE para símbolos gráficos junto con una instrucción READ ... DATA en el que los datos (no indicados) consistirán en un conjunto de números de direcciones y números de códigos de gráficos en pares.

Para símbolos gráficos más rápidos como los que se necesitan para animación, habrá que utilizar otros métodos con código de máquina.

```
15110 FOR N = TO K
15111 READ G1, G2
15112 POKE G1, G2
15113 NEXT N
15114 RESTORE
15115 RETURN
```

Variables

PASADAS	K, número de elementos o bloques (pixel)
	(G1,G2) pares de datos
LOCALES	N, contador
DEVUELTAS	ninguna

15120 Ejes de diagramas

Los caracteres gráficos pueden usarse en programas científicos o de negocios para dibujar ejes de gráficos y esta rutina está diseñada para dicho fin.

La subrutina debe alimentarse con valores para X, los números de caracteres por línea y en Y el número de líneas por cuadro. Es muy útil dejar algún espacio, usando menos que el número máximo para estas cantidades y poniendo un TAB (5) o instrucción antes de las instrucciones PRINT. Obsérvese que los códigos de gráficos son los códigos del TRS 80 para barras horizontales y verticales. Se debe reemplazar estos números de código por los que sirvan para el propio ordenador.

Si esta rutina se ejecuta sin que continúe por una sección de programa parte del dibujo o diagrama será cubierto por el letrero READY que aparece en la pantalla al final del programa. La subrutina se solicitará normalmente y luego se continúa por una sección de programa que imprime información en el diagrama o posiblemente permite al usuario introducir puntos del diagrama moviendo el cursor.

En cualquier caso, los ejes no serán interrumpidos por el mensaje READY y el único problema es el de hacer que la próxima línea impresa aparezca en el lugar correcto. Como es costumbre los ordenadores equipados con PRINT@ son más fáciles de usar pero los mismos métodos que se han empleado antes, basados en instrucciones TAB y elevación del cursor, puede también usarse.

```
15120 FOR N = 1 TO X
15121 REM X CARACTERES POR LINEA
15122 PRINT CHR$ (140);: REM BARRA HORIZONTAL
15123 NEXT N
15124 FOR N = 1 TO Y
15125 REM Y LINEAS POR PANTALLA
15126 PRINT CHR$ (191)
15127 REM BARRA VERTICAL
15128 NEXT N
15129 RETURN
```

Variables

PASADAS	X, caracteres por línea
	Y, línea por pantalla
LOCALES	N, contador
DEVUELTAS	ninguna

15130 Encontrar el máximo en una lista

Esto no es un programa de gráficos en el sentido de que produce una salida en la pantalla, sino una rutina muy útil para usar con las de dibujos de gráficos.

Las escalas de un diagrama a menudo tienen que ajustarse de modo que el mayor elemento de la lista se adapte al diagrama. Esta rutina encuentra el máximo valor almacenado tomando el primer elemento como el máximo y luego comparándolo con todo el resto de los elementos, intercambiando cuando sea necesario de modo que la variable KM contenga el número máximo al final de la rutina. El número de elementos en el vector debe traspasarse como variable N.

```
15130 KM = K(1)
15131 FOR Z = 2 TO N
15132 IF KM < K (Z) THEN 15135
15133 NEXT Z
15134 RETURN
15135 KM = K (Z)
15136 GOTO 15133
```

Variables

PASADAS	N, número de elementos en la lista
	K() vector de números
LOCALES	Z, contador
DEVUELTAS	KM, elemento de valor máximo

15140 Impresión de escalas en diagramas

Aquí se hace para un eje solamente, pero los principios son los mismos para el otro eje del diagrama.

El valor máximo para el vector que hay que expresar gráficamente KM obtenido de la subrutina previa sirve para dividir el eje X del diagrama en diez divisiones iguales marcando estos valores en los ejes. Se redondea para evitar el uso de fracciones. El número de caracteres por línea se debe traspasar a la subrutina como la variable X. Advertir que si hay que representar valores fraccionarios, se debería haber omitido el redondeamiento.

```
15140 Y = INT (KM/10) + 1
15141 FOR Z = 1 TO 10
15142 PRINT TAB (Z*X/10) INT (Y*Z);
15143 NEXT Z
15144 RETURN
15145 REM X CARACTERES POR LINEA
```

Variables

PASADAS	X, caracteres por línea
	K M elemento de mayor tamaño
LOCALES	Y,X, contadores
DEVUELTAS	ninguna

6

Archivo de datos en casete

El archivo de datos en casete no es una parte estandarizada del BASIC de modo que cada tipo de ordenador usa su propia versión de archivo de datos en casete. Algunas máquinas no tienen esta posibilidad por lo que no pueden usarse incluso para las aplicaciones más elementales en negocios o cálculos científicos en los que se manejan muchos datos. El problema de archivo de datos en casete ha sido que la mayor parte de los fabricantes lo han considerado útil; sin embargo, un usuario consciente utilizaría disquetes, algo que el usuario no se tomaría la molestia de tener porque usaría discos. La razón por la que una persona que compre un ordenador de £ 300, invertiría otras £ 100 en un sistema de discos no ha sido dada nunca, y se han necesitado los esfuerzos de fabricantes especializados para demostrar que son posibles los sistemas de archivo de datos de un modo rápido y fiable en casetes.

Quizás los fabricantes en el futuro (cuando Vd. lea esto) se habrán dado cuenta de las posibilidades de usar sistemas con casete para almacenamiento digital como el sistema minicasete. Philips que se usa en máquinas mayores.

Dejando esto por el momento, el sistema de utilizar tonos de audio grabados para transmitir los dígitos 0 y 1 es común a todos los sistemas con casete, pero el único estandarizado es el conocido como Kansas City o CUTS. En cualquier caso, aun si dos ordenadores utilizan el mismo método para codificar bits individuales en cinta, no hay garantía de que usarán el mismo método de codificar los datos en estos byts. La tendencia hoy es la de suministrar sistemas de archivo de datos en casete con opciones de alta velocidad usando tasas de 1200-1800 baudios contra los 300 baudios del estándar CUTS.

Ya que no hay forma estándar de archivo de datos, es gratificante que un gran número de máquinas use instrucciones similares. Estos son a menudo modelados según las instrucciones usadas para archivo con discos, de modo que el usuario pueda pasar de un modo más o menos ordenado de archivos de casetes a archivos de discos cuando lo necesite.

La excepción a este modelo es el TRS 80/Video Genie que utiliza un sistema de archivo de datos en casete más pobre cuando lo comparamos con otras máquinas.

De las nuevas máquinas, el sistema de archivo de datos en casete del micro BBC y de su antecesor el ACORN ATOM es con mucho el mejor.

Para muchos propósitos el sistema de archivo con casete del micro BBC sería tan útil al usuario como el sistema de archivo con discos de algunas máquinas que no voy a mencionar.

El mejor tipo de archivo de casetes necesita que se abra un archivo (usando OPEN o CREATE) clasificando los datos listos para grabar y luego grabados en cinta utilizando una instrucción como PRINT#1; el archivo debe luego cerrarse. La lectura consiste en abrir un archivo para preparar el almacenamiento y asignación del nombre de la variable usando INPUT#1 para leer la cinta, luego cerrando el archivo. Las operaciones OPEN y CLOSE se deben hacer en el sitio correcto y en el orden también correcto y un archivo se debería cerrar inmediatamente después de leer o escribir porque de no hacerlo así se pueden producir detenciones o convertir la cinta en indescifrable. Los microordenadores que no necesitan instrucciones OPEN o CLOSE son el TRS-80 y Video Genie.

16000 Advertencias

Los grabadores de casete, a diferencia de los equipos para discos, no están sometidos al control del ordenador, aunque algunos controlan el motor del grabador de la casete, parándolo y arrancándolo cuando se requiera.

Por esto, uno puede enfrentarse con la utilización de una instrucción de grabación de datos con el grabador de la casete desconectado y este desastre se debe evitar imprimiendo un mensaje de aviso o atención y parando el ordenador.

La subrutina imprime el correspondiente aviso de acuerdo con el valor de A$ (que debería ser GRABANDO o LEYENDO) que se transfiere a la subrutina.

La subrutina debería venir seguida por otra que requiera que se oprima una tecla antes de que se ejecute una instrucción de grabación para que así el usuario tenga tiempo para seleccionar la casete, encontrar el lugar adecuado y oprimir la tecla correcta en el grabador. Este sistema puede parecer tedioso comparado con la simplicidad de archivo con discos pero tiene la ventaja que es más difícil el cometer errores. Se necesita mucho más tiempo para sobre escribir una casete entera que un disco.

```
16000 CLS
16001 PRINT TAB (X) A$
16002 PRINT "PREPARE EL GRABADOR CON UNA CASETE"
16003 PRINT "POSICIONADA EN EL LUGAR CORRECTO"
16004 IF A$ = "GRABACION" THEN 16008
16005 PRINT "APRIETE LA TECLA PLAY DEL GRABADOR"
16006 PRINT "LUEGO TECLEE CUALQUIER LETRA EN EL
      ORDENADOR"
16007 RETURN
16008 PRINT "APRIETE LAS TECLAS RECORD Y PLAY DEL
      GRABADOR"
16009 GOTO 16006
```

Variables

PASADAS	X, número de tabulación del mensaje
	A$, mensaje (GRABACION O LECTURA)
LOCALES	ninguna
DEVUELTAS	ninguna

16010 Conservar o almacenar un vector

Esta rutina es la típica entre las que se usan para conservar datos en muchos ordenadores.

Se debe haber abierto primero el archivo y se debe también haberlo definido. La transferencia a la cinta se puede hacer después de haber hecho el aviso para preparar el grabador con el bucle contenido en las instrucciones 16012, 16013; después se cierra el archivo en 16014 y la rutina vuelve al programa que la solicitó. La variable K y el vector A$ deben suministrarse a la rutina.

La lectura sigue un procedimiento similar con el mismo tipo de bucle en conjunción con la instrucción PRINT # para leer los datos de la cinta. Advertir que algunas impresoras usan el signo de la libra en lugar del carácter sostenido (#)

```
16010 OPEN#10, "SALVAR": REM ALGUNAS MAQUINAS
      EMPLEAN 'CREATE # 10'
16011 FOR N = 1 TO K
16012 PRINT # 10,A$ (N)
16013 NEXT N
16014 CLOSE # 10
16015 RETURN
16016 REM EMPLEAR EL MISMO BUCLE PARA LEER
```

Variables

PASADAS	K, número de elementos
	A$() vector de elementos
LOCALES	N, contador
DEVUELTAS	ninguna

16020 Archivo con el TRS/80 - Video Genie

El sistema de este ordenador usa las instrucciones PRINT #—1 e INPUT#—1 y no tiene las OPEN y CLOSE. Es probablemente más fácil de usar para el usuario comparado con un sistema más sofisticado de archivo, porque la instrucción PRINT #—1 viene seguida por una lista separada por comas de las variables cuyos valores hay que conservar. Con tal de que el número de bytes conservados por una instrucción PRINT#—1 no exceda de 248, el sistema es realmente muy simple, como se describe en la pareja de subrutinas para escribir y leer datos.

Advertir que se han usado los mismos nombres de variables, aunque no sea necesario, pero el mismo orden de tipos de variables se debería usar siempre.

```
16020 PRINT # —1, A,B,C, A$, B$
16021 RETURN
16022 REM EMPLEAR LA MISMA ORDEN PARA LEER
16023 INPUT #—1, A,B,C, A$, B$
16024 RETURN
16025 REM EL TRS-80 EMPLEA ESTE METODO
```

Variables

PASADAS	A,B,C, números
	A$ B$ tiras
LOCALES	ninguna
DEVUELTAS	ninguna

16030 Agrupamiento de tiras para el TRS-80/Video Genie

Los problemas con el archivo de datos del TRS/80 Video Genie surgen cuando se tienen que grabar vectores o matrices. Cada llamada a la rutina PRINT#—1 en un bucle hará que arranque el motor que gire a la velocidad normal e imprima 250 bytes de ceros seguidos por un byte de sincronización y posteriormente los datos. Esto se hace incluso si hay sólo un byte de datos por llamada. Es imposible sin escribir total o parcialmente un nuevo sistema operativo para casete mantener la cinta desplazándose y grabar todos los valores de un vector consecutivamente como se hace normalmente en el resto de las máquinas. Algunas modificaciones para lograr mayor velocidad (de hardware y software) permiten una mayor eficiencia en el archivo de datos, en conjunción con una mayor velocidad de carga y descarga a menudo mayor que la que se logra con otras máquinas (el equipo TC 8 para el TRS 80 es uno de ellos).

Algunos usuarios han escrito también rutinas en lenguaje de máquina que reemplazan la larga parte delantera por otra más corta cuando se está grabando un vector numérico.

Esta rutina es mi propia solución al problema. Los datos se ponen en la forma de vector de tiras y luego concatenadas en tiras largas cada una de las cuales se graba; para leerlas se separan en un vector de nuevo.

La mayor parte de las rutinas que comprimen y descomprimen son lentas, lo que tiende a disminuir la ventaja del método. Este método es más rápido que la mayoría. El principio se basa en medir la longitud de cada tira, convirtiendo este número a una tira, rellenándola y seccionándola hasta obtener una longitud fija (línea 16034) y después concatenando la longitud de la tira y la tira misma en una tira más larga que consistirá en un conjunto de números y letras.

Los números son las longitudes de las tiras siguientes que es la señal para el procedimiento de separación.

Si la longitud de la tira comprimida alcanza un valor que es demasiado grande para que el sistema operativo la maneje (MAX) entonces se graba, se borra y se continúa con la compresión. Antes de que se graben las tiras, sin

embargo, el número total de tiras en el vector se convierte en tira y se graba separadamente para permitir después una separación eficiente. Esta rutina es mucho más rápida para distribuciones numéricas y tiras que el método estándar y hace que el archivo de datos en estas máquinas sea más efectivo sin tener que recurrir a rutinas con código de máquina.

```
16030 A$ = STR$ (K)
16031 GOSUB 16043
16032 A$ = ""
16033 FOR N = 1 TO K
16034 C$ = RIGHT$ ("" + STR$ (LEN (B$(N))),2)
16035 A$ = A$ + C$ + B$ (N)
16036 IF LEN (A$) > MAX THEN 16040
16037 NEXT N
16038 GOSUB 16043
16039 RETURN
16040 GOSUB 16043
16041 A$ = ""
16042 GOTO 16037
16043 PRINT #—1,A$
16044 RETURN
```

Variables

PASADAS	K, número de elementos en el vector
MAX	longitud máxima de la tira
	B$ () vector de tiras
LOCALES	N, contador
	C$ código de longitud de la tira
DEVUELTAS	A$ tira comprimida para grabación

16040 Descompresión de tiras para formar un vector

Esta es la rutina de la descompresión que se corresponde con la rutina de agrupamiento 16030. Empieza leyendo la tira cuyo valor da el número total de elementos K. El número de cuenta N se pone a 1 y la primera tira agrupada se lee en la línea 16043. El número de posición J se pone a 1.

En la línea 16045 se determina un número que es la longitud del primer elemento o tira de la distribución siendo el valor de los dos primeros caracteres de la tira. Este número se usa luego en 16046 para obtener la tira y

asignarla en la distribución. Los números de cuenta se examinan para comprobar que no se ha alcanzado el final de la tira y se aumentan en 16048 y 16049 para poder leer y extraer la siguiente tira del vector de la tira agrupada retornando a 16045.

Cuando se llega al final de la tira comprimida, se lee otra tira de la cinta y se continúa el proceso hasta que todos los elementos especificados por la variable K se han asignado al vector sobre el que vuelve la subrutina. Esta descompresión es mucho más rápida que la que se describió anteriormente con la descompresión carácter por carácter utilizando el carácter ASCII 128 como un separador.

```
16040 GOSUB 16054
16041 K = VAL (A$)
16042 N = 1
16043 GOSUB 16054
16044 J = 1
16045 X = VAL (MID$(A$,J,2))
16046 B$ (N) = MID$ (A$,J+2,X)
16047 IF J + X + 2 >= LEN (A$) THEN 16051
16048 N = N+1
16049 J = J+X+2
16050 GOTO 16045
16051 N = N + 1
16052 IF N < K THEN 16043
16053 RETURN
16054 INPUT #—1,A$
16055 RETURN
```

Variables

PASADAS	(indirectamente) A$ tira grabada
LOCALES	K,N,J,X, contadores
DEVUELTAS	K, número de elementos
	B$() vector de tiras

16050 Validación

Es muy útil leer datos de la cinta después de su escritura mientras que los datos originales están aún almacenados, de modo que la lectura de datos es doblemente comprobada. Esta subrutina compara un vector grabado y leído posteriormente B$ con una A$ que se retiene en la memoria e informa sobre

cualquier elemento que no concuerde. Tiene que utilizarse cuidadosamente porque cuando se graba un número y se lee después, volverá en algunas máquinas precedido de un espacio en blanco que pueda originar un informe de error.

```
16050 FOR N = 1 TO K
16051 IF A$ (N) < > B$ (N) THEN 16054
16052 NEXT N
16053 RETURN
16054 PRINT "ITEM"; N; "ES ERRONEO——"
16055 PRINT A$ (N) "SE GRABO" B$(N) "SE LEYO"
16056 GOTO 16052
```

Variables

PASADAS	K, número de elementos
	A$() vector de elementos en memoria
	B$() vector de elementos de la cinta
LOCALES	N, contador
DEVUELTAS	ninguna

16060 Copia de datos

Los datos almacenados en una casete no pueden cambiarse fácilmente excepto, leyéndolos, cambiándolos y escribiéndolos de nuevo. Además hacer una cinta con copia de datos también necesita una rutina para leer y escribir. Esta subrutina ofrece una limitada habilidad para leer y escribir. Está preparada para el TRS-80 Video Genie pero podría usarse con las instrucciones OPEN y CLOSE con otros sistemas de archivos de datos. El principio es simplemente leer un conjunto de datos, grabando en otra cinta y luego borrando las variables. A menos que se use un sistema de dos casetes, el proceso será lento y se debe tener cuidado para que la lista de variables que se lee en 16061 corresponda a la secuencia que es realmente grabada en la cinta.

```
16060 GOSUB 16068
16061 INPUT #—1,A$,B,C
16062 GOSUB 16070
16063 PRINT #—1,A$,B,C
```

```
16064 A$ = ""
16065 B = 0
16066 C = 0
16067 RETURN
16068 REM AVISO DE RELECTURA
16069 RETURN
16070 REM AVISO DE ESCRITURA
16071 RETURN
```

Variables

PASADAS	ninguna (datos grabados)
LOCALES	A$,B,C, elementos de datos
DEVUELTAS	ninguna (datos grabados)

Apéndice A

Códigos que no se imprimen

N.º	TRS-80 VIDEO GENIE	RML 380-Z	ACORN ATOM	APPLE II	BBC MICRO
1					Salida para impresora solamente, no para pantalla
2			Empezar la impresora	Restaurar	Conectar la impresora
3			Para la impresora	Paso	Desconectar la impresora
4		Resumen de salida			Separar el cursor del texto, del cursor de los gráficos
5					Unir ambos cursores
6			Salida a pantalla		Conectar el VDU
7			Beep (zumbido)	Beep (zumbido)	Beep (zumbido)
8	Retroceso y borrado	Retroceso	Retroceso	Retroceso	Retroceso
9		Tabulación	Adelanto		Adelanto
10	C/R, L/F	L/F	L/F	L/F	L/F
11		Línea arriba	Línea arriba		Línea arriba
12		Borrar, cursor a rincón izdo. inferior	Borrar, cursor a punto de partida		Borrar, cursor a punto de partida
13	C/R, L/F	C/R, L/F	C/R	C/R	C/R
14	Cursor conectado	C/R	Formar página en la pantalla		Páginas conectadas
15	Cursor desconectado	Supresión de salida	Páginas desconectadas		Páginas desconectadas
16					Borrar el área de gráficos

Códigos que no se imprimen (continuación)

N.º	TRS-80 VIDEO GENIE	RML 380-Z	ACORN ATOM	APPLE II	BBC MICRO
17		Borrar cuenta de páginas			Definir color del texto
18		Establecer la cuenta de páginas			Definir color y gráficos
19				Control del desplazamiento	Definir color de la lógica
20					Restaurar el color del valor defectuoso
21		Titilar	Sin salida a pantalla	Cursor hacia adelante	Desconexión de VDU
22		Modificación del cursor			Modo para seleccionar la pantalla
23	32 caracteres por línea en TRS 80	Destellos desconectados			Reprogramar la visualización de caracteres
24	Retroceso del cursor	Cursor a la derecha	Cancelar línea	Borrar línea	Definir ventana de gráficos, Instrucción para dibujar
25	Adelanto del cursor	Borrar hasta el fin de línea			
26	Avance de línea hacia abajo				Restaurar tamaño de la ventana por defecto
27	Desplazamiento de línea hacia arriba				Sin acción
28	Cursor al punto inicial				Definir ventana de texto
29	Cursor al principio de línea	Cursor al punto de origen			Definir origen de gráficos
30	Borrado hasta el fin del del cuadro	Borrado hasta el fin de línea	Cursor al punto de origen		Cursor del texto a lugar de origen
31	Borrado hasta fin del cuadro	Borrar la pantalla cursor al punto de origen			Mover cursor del texto a nueva posición
		(N. B. CHR$ (255) = retroceso y borrado			(N. B. CHR$ (127) = retroceso y borrado

Apéndice B

SUBRUTINAS EN CODIGO MAQUINA

Una ventaja de escribir programas en BASIC en la forma de un pequeño núcleo que contine un menú u otros procedimientos de acceder a subrutinas, es que todas las acciones importantes del programa son entonces realizadas por las subrutinas.

Esto hace posible alterar un programa muy fácilmente cambiando la subrutina relevante y si la numeración de las subrutinas se han elegido bien, es posible hacerlo sin tener que volver a numerar todo el programa.

Algunos tipos de programas requerirán subrutinas en código de máquina para ejecutar acciones que o bien no están disponibles en BASIC o son muy lentas cuando se usa este lenguaje. Esto puede evitarse en algunas máquinas si se dispone de un compilador para transformar el BASIC en una mezcla de BASIC y código de máquina. Los usuarios del TRS 80 tienen la opción de usar el grupo de compiladores *Southern Software's ACCEL* que acelerará notablemente la ejecución de un gran número de instrucciones en BASIC. Esto por supuesto acelerará todas las acciones, de modo que los bucles FOR ... NEXT que se han utilizado para temporización, tendrán que cambiarse antes de que la versión final del programa se compile se pueden necesitar varias ejecuciones de prueba para que quede bien.

En la mayor parte de los ordenadores sin embargo, la única opción de que se dispone para lograr que una subrutina sea más rápida, suponiendo que el programa en BASIC se ha hecho tan eficiente como sea posible y que se han hecho todos los esfuerzos necesarios para eliminar procedimientos que desperdician el tiempo, como recoger la información inútil, es acudir a escribir la subrutina en código de máquina.

Escribir códigos de máquina de una manera eficiente constituye un logro en sí mismo, pero incluso la rutina más ineficiente en código de máquina es probablemente mucho más rápida que en BASIC para la mayor parte de los propósitos.

La ventaja de una mayor velocidad en máquinas tales como SPECTRUM puede ser asombrosa; por otro lado, el uso del BASIC en el microordenador BBC es tan rápido que utilizar el código de máquina no vale realmente la pena, si de lo que se trata únicamente es de obtener mayor velocidad.

En este apéndice, sin embargo, nos ocuparemos de como subrutinas en código de máquina pueden usarse con programas en BASIC, más bien que de la mecánica de escribir tales rutinas de las que tratan textos más especializados.

Para empezar, los bytes en código de máquina deben colocarse en la memoria donde no puedan corromperse o estropearse por la acción normal del ordenador. El ordenador hace un uso considerable de la memoria durante la ejecución de un programa para almacenar los valores de variables, el texto de las tiras y para el almacenamiento temporal de una cantidad mientras se usa.

Normalmente, el ordenador preservará sólo la parte de la memoria en la que está almacenado el programa en BASIC y usará el resto de la memoria cuando se necesite. En general, las direcciones de memoria justamente encima de las usadas, y las que están cerca de la parte superior se utilizan de esta manera. Algunos ordenadores permiten que un área en la parte superior de la memoria se reserve para programas en código de máquina usando una instrucción en BASIC. En el SPECTRUM, por ejemplo, la instrucción CLEAR seguida por un número de dirección de memoria (típicamente 32500) reservará un espacio en la memoria sobre este número. Este espacio se extiende a la dirección en la que se almacenan los caracteres definidos gráficamente por el usuario (ver el manual SPECTRUM para detalles). El TRS 80 puede también reservar una parte de su gran número de direcciones de memoria para rutinas en código de máquina y el método convencional de hacer esto es imprimir el número de memoria requerido *al conectar* cuando el mensaje MEM SIZE? o (MEMORY SIZE? para los modelos anteriores) aparece en pantalla. Como antes, esto reserva direcciones de memoria de este valor y más altos para las rutinas en código de máquina. Es a menudo poco conveniente tener que volver al procedimiento de conectar y es posible introducir (con POKE) el valor de la dirección entre dos direcciones en la memoria, ejecutar una instrucción CLEAR en BASIC y así establecer el tamaño de la memoria automáticamente dentro de un programa cuando éste empieza a ser ejecutado.

Esto no debe hacerse, excepto como la primera instrucción en un programa; de otro modo el programa será inservible porque el ordenador habrá ya hecho algún uso de la memoria que ha sido ahora reservada.

Un método mucho menos documentado de asignar espacio para subrutinas en código de máquina es cambiar el punto de arranque del BASIC. La dirección de arranque para el BASIC se mantiene generalmente en dos bytes de memoria reservada y alterando estos dos bytes, la dirección de arranque puede llevarse a una dirección más alta de memoria. Esto permite que se cargue un programa en BASIC desde la cinta a un conjunto más alto que el normal de direcciones, dejando los números de direcciones debajo del arranque en BASIC, libres para el código de máquina ya que la máquina no usará estas direcciones.

Este método puede adaptarse para prácticamente cualquier tipo de ordenadores y vale la pena investigarlo porque el código de máquina puede cargarse antes o después del programa en BASIC, con tal que la memoria haya sido asignada correctamente. Un método muy elegante es cargar el código de máquina, estableciendo la auto ejecución, de modo que una parte del código de máquina reasignará el espacio para BASIC, y entonces empezará la carga del programa en BASIC. El código BASIC entonces se escribe sobre esta sección del código de máquina que no se necesita ya, pero deja la otra sección que contiene la subrutina.

Este método también tiene la ventaja de que es muy difícil copiar el programa; una alternativa es cargar y ejecutar un programa en BASIC que reasigna la memoria, carga el programa principal en BASIC y luego carga el código de máquina escribiendo sobre la sección de carga del BASIC.

Ejemplos de esta técnica se encuentran en varias cintas comercialmente disponibles.

Otro método de proteger el código de máquina para que no sea afectado por el sistema operativo del ordenador es colocar los bytes del código máquina en instrucciones REM o en tiras o tablas.

El primer método es el tradicionalmente empleado para colocar el código de máquina en la serie de ZX80/ZX81 escribiendo las REM seguidas por tantos espacios como bytes de código máquina haya que poner.

Al hacer esta instrucción REM la primera en el programa se fija su dirección de memoria y el código máquina puede colocarse en su lugar. Un método similar puede utilizarse con el SPECTRUM con la ventaja que las instrucciones READ ... DATA pueden usarse para llenar los espacios con bytes. El uso de tiras o de tablas de números es un método favorecido por los programadores TRS-80. La dirección de arranque de una variable almace-

nada en la memoria puede encontrarse utlizando la utilísima instrucción VARPTR de esta máquina.

La desventaja de usar tiras para almacenamiento es que la dirección de una tira en la memoria no está fijada, de modo que la instrucción VARPTR se debe usar para encontrar la posición de la tira antes de que se utilice. Es preferible usar un vector numérico para contener los bytes de un programa en código máquina. Debe haber por supuesto algún método de introducir los bytes, y esto se hace generalmente bajo la forma de instrucciones del tipo READ ... DATA.

Cuando el código de máquina se localiza en la memoria reservada, puede cargarse desde una grabación separada, que puede estar en la misma cinta que el programa BASIC o puede estar contenido dentro del programa BASIC como se indica con instrucciones DATA, o en el caso de la serie ZX con instrucciones REM que pueden grabarse con el resto del BASIC.

Llamada a la subrutina en código máquina

Los métodos que se usan para solicitar una subrutina en código máquina varían considerablemente de un ordenador a otro, incluso cuando la forma de la instrucción parece que es idéntica. La palabra más común para llamar la subrutina es USR seguida por un número pero la sintaxis no es uniforme dependiendo de lo que represente el número. Algunas máquinas especialmente SPECTRUM, interpretan el número como una dirección, de modo que una instrucción tal como PRINT USR 32500 hará que se ejecute la rutina en código de máquina que empieza en la dirección 32500. USR puede colocarse junto con PRINT asignado como LET x= USR 32500 o usar la variable en una instrucción RANDOMIZE tal como RANDOMIZE 32500. La ventaja de usar este método de solicitar o llamar a una subrutina es que así resulta muy fácil probar las rutinas, incluso rutinas en la ROM del ordenador, aplicando las rutinas USR. El método PRINT USR dará como resultado la impresión de un número en la pantalla, el utilizar LET x = USR (dirección) asignará un número a la variable x; solamente PAND USR (dirección) no producirá ningún efecto en el programa principal. SPECTRUM organiza el sistema USR de manera que el número presente en el par de registros BC del micro Z-80A será devuelto por la instrucción USR y éste es el número que aparecerá si se usa el método PRINT USR, o asignado a x en LET x = USR (dirección). Si los registros BC no se cambian debido a la subrutina en código de máquina, entonces el número que se devuelve es simplemente la dirección que seguía a USR.

Aunque este esquema es muy simple, tiene sin embargo la desventaja que no transfiere un valor a la subrutina. Esto puede ser compensado escribiendo en la memoria el valor que tiene que transferirse y gestionando la subrutina para que use esta parte de la memoria. Una alternativa es escribir una porción de la subrutina de modo que lea el valor de una o más variables a las que se han asignado valores por el programa BASIC. Esto es muy parecido al método usado en la máquina BBC.

Otra variación de USR encontrada específicamente en MICROSOFT BASIC sigue a la llamada de URS con un número entre paréntesis. Este número es el que se transfiere a la subrutina en código de máquina para que se ponga en los registros del microprocesador, y la dirección a la que la rutina URS dirige al ordenador tiene que escribirse en la memoria, en vez de formar parte de la instrucción. Una llamada de esta clase devolverá también un valor derivado de los registros de los microprocesadores.

La mejor provisión para subrutinas en código de máquina, es como debiera esperarse la de la máquina BBC. Ambas, direcciones y valores pueden especificarse, los valores se colocan como "variables enteras residentes" y la dirección se especifica en la instrucción CALL. La simplicidad de mezclar el código de máquina con el BASIC hace que esta máquina destaque para todo tipo de propósitos.

OTROS LIBROS DE INFORMATICA DE

Diccionarios

HART.– Diccionario del Basic. (2ª Edic.) 1985.

NANIA.– Diccionario de Informática. Inglés. Francés. Español. Más de 11000 términos. 1985. Rústica.

OLIVETTI.– Diccionario de Informática. Inglés-Español (7ª Edic.) 1986.

Hardware (Equipo físico)

ANGULO.– Electrónica digital moderna. (6ª Edic.) 1985.

ANGULO.– Memorias de burbujas magnéticas. Tecnología y sistemas de aplicación. 1982.

ANGULO.– Microprocesadores. Arquitectura, programación y desarrollo de sistemas (3ª Edic.) 1984.

ANGULO.– Microprocesadores. Curso sobre aplicaciones en sistemas industriales. (4ª Edic.) 1985.

ANGULO.– Microprocesadores. Diseño práctico de sistemas. (3ª Edic.) 1986.

ANGULO.– Microprocesadores. Fundamentos, diseño y aplicaciones en la industria y en los microcomputadores. (4ª Edic.) 1985.

ANGULO.– Microprocesadores de 16 Bits. El 68000 y el 8086/8088. (2ª Edic.) 1985

ANGULO.– Prácticas de microelectrónica y microinformática. (2ª Edic.) 1985.

ASPINALL.– El microprocesador y sus aplicaciones. 1984.

GARLAND.– Diseño de sistemas microprocesadores (2ª Edic.) 1984.

HALSALL.– Fundamentos de microprocesadores. 1984.

LEPAPE.– Programación del Z80 con ensamblador. 1985.

ROBIN.– Interconexión de microprocesadores. (2ª Edic.) 1985.

RONY.– El microprocesador 8080 y sus interfases. 1984.

Lenguajes

BELLIDO Y SANCHEZ.– Basic para estudiantes. 1985

BELLIDO Y SANCHEZ.– Basic para maestros. (2ª Edic.) 1985

BELLIDO, SANCHEZ Y RODRIGUEZ.– Casete con los programas de Basic para Maestros. 1985.

BURKE.– Enseñanza asistida por ordenador. 1986.

CUÑAT.– Pascal para estudiantes. 1986.

CHECROUN.– Basic. Programación de microordenadores. (5ª Edic.) 1985.

DELANOY.– Ficheros en Basic. (3ª Edic.) 1986.

GALAN PASCUAL.– Programación con el lenguaje Cobol. (5ª Edic.) 1986.
GARCIA MERAYO.– Programación en Fortran 77. 1986.
LARRECHE.– Basic. Introducción a la programación. (5ª Edic.) 1985.
MARSHALL.– Lenguajes de programación para Micros. 1985.
MONTEIL.– Primeros pasos en Logo. (2ª Edic.) 1985.
OAKEY.– Lenguaje Forth para micros. 1985.
QUANEAUX.– Tratamiento de textos con Basic. 1985.
RAMON.– 44 superprogramas en Basic. 1985.
ROSSI.– Basic. Curso acelerado. (5ª Edic.) 1985.
SANCHIS Y MORALES.– Programación con el lenguaje Pascal. (5ª Edic.) 1985.
WATT Y MANGADA.– Basic para niños (7ª Edic.) 1986.
WATT Y MANGADA.– Basic avanzado para niños. (3ª Edic.) 1985.

Aplicaciones e informática profesional

ABRAMSON.– Teoría de la información y codificación. (5ª Edic.) 1981.
ANGULO.– Guía fácil inteligencia artificial. 1986.
BUITELAAR.– Informática y medicina. 1986.
COHEN.– Estructura lógica y diseño de programas. 1986.
DAX.– CP/M. Guía de utilización. MP/M, CP/M 86, Wordstar, Basic. 1986.
FLORES.– Estructuración y proceso de datos. (5ª Edic.) 1985.
GAUTHIER.– Diseño de programas para sistemas. (5ª Edic.) 1985.
HARTMAN.– Manual de los sistemas de información. T.1. (7ª Edic.) 1985.
HARTMAN.– Manual de los sistemas de información. T.2. (7ª Edic.) 1985.
LANE.– Telemática y comunicaciones en la empresa, 1986.
LEWIS.– Estructuras de datos. Programación y aplicaciones. 1985.
LUCAS.– Sistemas de información. Análisis. Diseño. Puesta a punto. 1984.
POPKIN.– Introducción al proceso de datos. 1986.
RODRIGUEZ.– Protección de la información. Diseño de criptosistemas informáticos. 1986.
SANCHIS.– Teoría y construcción de compiladores. 1986.
SCHMIDT.– Introducción a los ordenadores y al proceso de datos. (5ª Edic.) 1985.

GUIA FACIL DE SUBRUTINAS UTILES EN BASIC

La mayor parte de los programas para computadores a pesar de que tengan claras diferencias, usan solamente un grupo limitado de acciones sobre los datos. Si éstas acciones se pueden ejecutar con rutinas estándar (subrutinas) entonces se puede ahorrar mucho trabajo a la hora de programar.

GUIA FACIL DE SUBRUTINAS UTILES EN BASIC, contiene listados, notas aclaratorias y listas de variables para muchas subrutinas, entre las que se incluyen, intermitencia del título, impresión en columnas, visualización gráfica enmarcada, título desplazable, subrayado, clasificación, etc.

Aunque la mayoría de las subrutinas sirven para casi todas las máquinas, o se dan las instrucciones necesarias para su adaptación a la máquina del lector, se han incluido unos cuantos ejemplos de rutinas que pretenden superar limitaciones o aprovecharse de características de ciertas máquinas.

Entre ellas figuran rutinas para ayudar al archivo en casete en el TRS-80/Video Genie y tablas para mostrar las versiones de las órdenes de seccionamiento de cadenas usando MICROSOFT para los Micros SPECTRUM, ZX81 y ACORN ATOM.

Todas las subrutinas tienen comentarios con respecto a las variables globales y locales para que puedan adoptarse como Procedimientos para el micro-computador BBC.

EDITORIAL PARANINFO S.A.

Magallanes, 25 - 28015 Madrid

ISBN: 84-283-1465-9